(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 03/021447 A1

(51) International Patent Classification[7]: G06F 12/00, H04L 12/56

(21) International Application Number: PCT/US02/27709

(22) International Filing Date: 30 August 2002 (30.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/316,651 31 August 2001 (31.08.2001) US

(71) Applicant *(for all designated States except US)*: **ADAPTEC, INC.** [US/US]; 691 South Milpitas Blvd., Milpitas, CA 95035 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VENKATARAMAN, Srinivasan** [IN/US]; 699 Patriot Place, Fremont, CA 94539 (US). **JAYAM, Ramkumar** [IN/US]; 4573 Terra Place, San Jose, CA 95121 (US). **MUNNANGI, Sivakumar** [IN/US]; 2266 Lenox Place, Santa Clara, CA 95054 (US). **KAPATKAR, Anil** [IN/US]; 4719 Blue Ridge Dr., San Jose, CA 95129 (US).

(74) Agent: **NGUYEN, Joseph, A.**; Ipsg, P.C., P.O. Box 61147, Sunnyvale, CA 94088 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

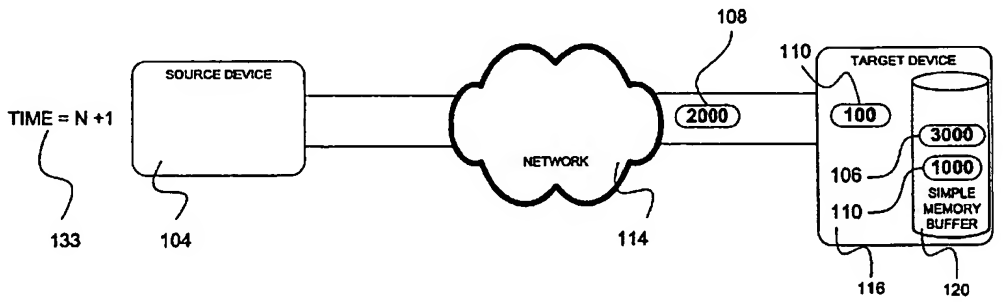
Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY,*

[Continued on next page]

(54) Title: METHODS AND APPARATUS FOR PARTIALLY REORDERING DATA PACKETS



(57) Abstract: A method in a target device (116) for partially reordering a plur ality of data packets transmitted from a source device (104). The source device (104) is coupled to the target device (116) viaa computer network (114). The method of receiving a first set of data packets from the transmitted device (104), and ascertaining whether the first set of data packets represents a set of data packets that the target device (116) expects to receive next. If the first set of data packets does not represents the set of data packets that the target expects to receive next, the method includes storing the first set of data packets in a memory buffer (120) of the target device (116). The storing includes arranging the first set of data packets in the memory buffer (120) such that data packets in the memory buffer, including the first set of data packets (120), are in order in the memory buffer (120).

WO 03/021447 A1

BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHODS AND APPARATUS FOR PARTIALLY REORDERING DATA PACKETS

BACKGROUND OF THE INVENTION

1 The present invention relates in general to networking technologies and in particular to methods and apparatus for optimizing the transfer of data packets across a network.

2 Data traffic on the Internet continues to grow at phenomenal rates. Initially designed as low-bandwidth text medium, Internet traffic is now including greater amounts of high-bandwidth traffic, such as pictures, audio & video streaming, video conferencing, and online gaming.

3 The primary protocol suite of the Internet is TCP/IP, or Transmission Control Protocol/Internet Protocol. TCP/IP is actually two separate protocols that work together, a higher level protocol called Transmission Control Protocol, and a lower level protocol called Internet Protocol. TCP manages the segmentation of a message, file, or data stream (generically "file") into smaller packets, or segments, that are transmitted across the network from a source device to one or more target devices. When the packets arrive at their destination, packets that are in-order are reassembled with previously received packets, and an acknowledgement is sent back to the source device. IP, on the other hand, handles the address part of each packet so that it arrives at the proper target device.

4 For example, an application on a source device, such as a web server, may use TCP/IP to transmit information to an application on a target device, such as a web browser. However, since the Internet is connectionless, there is no guaranteed pre-determined path from one device on the network to another. There is, therefore, a very high probability that some packets from the same session will be routed differently than others, arriving at their destination in a different order from the order with which they were initially transmitted.

5 During the initial setup process, prior to the actual transmission of data, TCP/IP adjusts the size of the transmitted packet according to network conditions between the source and target device. If the packet is too small, the network will be unnecessarily congested, and the data will arrive at its destination needlessly late. If the packet size is too large, it will not arrive at its destination at all, and no acknowledgements will be sent back to the source device. If no acknowledgement is received for some time, TCP/IP may reduce the size of the packet and the number of packets sent until acknowledgements are received. Once acknowledgements are received, TCP/IP will begin to increase the packet size and number of packets again to attempt to optimize the data transfer.

6 In order to account for the variability of packet size, TCP/IP mandates that each packet contain a sequence number and a length. The sequence number is used to order the packets. Generally speaking, the sequence number of the packet corresponds to the order number of the first data byte in the packet within the total transmitted file. For instance, if a file of 2000 bytes were transmitted, and the 2nd packet transmitted has a sequence number of 1000, then 999 bytes have already been sent target device in the first packet.

7 The length is used to determine the sequence number of the next packet. For instance, if a file of 2000 bytes were transmitted, and the 2nd packet transmitted has a sequence number of 1000, and a length of 500, then the sequence number of the next packet is 1500.

8 In general, since applications can only assemble and interpret data in the correct sequential order, the target device only forwards to a target application the next expected sequential packet, along with other sequentially contiguous packets located in the memory buffer of the target device.

9 Fig. 1A shows, in accordance with a prior art technique, a simplified diagram showing a source device 104 transmitting a group of sequenced packets to a target device 116 across a network. In the example of Fig. 1A, target device 116 does not have a memory buffer for any incoming packets. Received packets whose sequence numbers are out of order are simply discarded. The source device 104 simply

retransmits the discarded packets to the target device 116, if acknowledgements are not received after a certain period of time.

10 At time=N (102), source device 104 transmits packet 100 (112), packet 1000 (110), packet 2000 (108), and packet 3000 (106) in sequence, to target device 116, across network 114. Packet 100 (112), with the smallest sequence number that has not been received, is the next expected packet at target device 116.

11 At a later time=N+1 103, packet 1000 (110) reaches target device 116 prior to the packet 100 (112). Target device 116, having no data buffering mechanism, simply discards packet 1000 (110), and waits for packet 100 (112), which in this example is arriving next. Source device 104, receiving no acknowledgement after a certain period of time from target device 116, simply retransmits packet 1000 (110).

12 Although relatively easy to implement, this method is problematic because it needlessly floods the network with retransmitted data packets. Not only will the network appear sluggish to the target application, since it must wait an additional amount of time to receive correctly sequenced packets, but the additional traffic will also reduce the overall performance of the network for all other connected devices by increasing congestion.

13 Referring now to Fig 1B, another prior art technique is shown in which the target device 116 contains a memory buffer 120. It is simple in the sense that packets are stored in vacant locations in no particularly order. Once the memory buffer 120 is full, any additional packets, other than the next expected packet, are discarded. The next expected packet is forwarded to the target application once it arrives at the target device, or once it is located in the memory buffer 120. A new next expected packet is then determined.

14 At time=N (132), source device 104 transmits packet 100 (112), packet 1000 (110), packet 2000 (108), and packet 3000 (106) in proper sequence, to target device 116, across network 114. Packet 100 (112) is the next expected packet at target device 116.

15 At a later time=N+1 133, packet 1000 (110) and packet 3000 (106) reach target device 116 prior to the packet 100 (112). Target device 116, having a memory buffer 120, places packet 1000 (110) in the first available slot, and packet 3000 (106) in the next available slot, and waits for packet 100 (112), which in the example of Fig. 1B is arriving next.

16 Once packet 100 (112) arrives, it is forwarded to the target application on the target device 116, and a new next expected packet is determined. The target device 116 then scans the entries in the memory buffer 120 for the new next expected packet. If it is located, it too is forwarded to the software application. In this diagram, packet 1000 (110) is the new next expected packet, and has already arrived at the memory buffer 120. The target device 116 would locate and forward packet 1000 (110) to the software application.

17 Although the use of a memory buffer 120 is an improvement over the implementation of Fig 1A, there are still disadvantages. The memory buffer 120 still transfers packets in an inefficient manner to the application, since the sequence number of each packet must be continuously re-inspected in the buffer when a new packet arrives at the target device. This re-inspection increases the network latency for the application by slowing the transfer rate of data packets..

18 When two computers communicate via a computer network using the TCP/IP protocol, a data structure known as a transmission control block (TCB) is typically employed to facilitate data transmission, segmentation, reassembly, retransmission, acknowledgement, and the like of datagrams in the bi-directional data flow between the communicating computers. The TCB is employed to track various parameters associated with the data transmit and receive process for a given data flow. Generally speaking, there is one transmission control block per data flow at each host computer system (i.e., the computer system involved in the communication at each end of a data flow). Each TCB is uniquely identified by its TCP source port, TCP destination port, IP source address, and/or IP destination address.

19 In the prior art, a transmission control block in a host computer is employed, for a given data flow, to facilitate both the transmission of data from that host

computer and the receiving of data into that host computer. Fig. 1C illustrates this situation wherein a TCB 162 is employed to facilitate both the transmission and the receiving of data for host computer 164. Likewise, a TCB 166 is employed to facilitate both the transmission and the receiving of data for host computer 168. However, if a transmission control block is busy servicing a transmission request in a host computer, it is unavailable for use to facilitate the receiving of data in that host computer until the transmit task is finished. Accordingly, a bottleneck exists which limits the transmission bandwidth between the two host computers.

20 If the data transmission speed between host computer 164 and host computer 168 is relatively low compared to the speed at which data is processed within the host computers, this bottleneck may be tolerable. As the transmission bandwidth between host computers increase, this bandwidth bottleneck increasingly becomes a critical issue. As bandwidth approaches 1Gbits/sec, 10Gbits/sec, or even higher for enterprise networking, and up to 40Gbits/sec or higher among network routers, it is clear that the bandwidth bottleneck needs to be resolved if TCP/IP is to remain a viable protocol suite for networking going forward.

21 In view of the foregoing, there is desired improved methods and apparatus for relieving the bandwidth bottleneck associated with the prior art transmission control block and for improving the data transmission speeds when two or more computers communicate using the TCP/IP protocol via a computer network.

SUMMARY OF THE INVENTION

22 The invention relates, in one embodiment, to a method in a target device for partially reordering a plurality of data packets transmitted from a source device. The source device is coupled to the target device via a computer network. The method receiving a first set of data packets from the transmitted device, and ascertaining whether the first set of data packets represents a set of data packets that the target device expects to receive next. If the first set of data packets does not represents the set of data packets that the target expects to receive next, the method includes storing the first set of data packets in a memory buffer of the target device. The storing includes arranging the first set of data packets in the memory buffer such that data packets in the memory buffer, including the first set of data packets, are in order in the memory buffer.

23 In another embodiment, the present invention relates to memory buffer structure and associated logic in a target device for partially ordering packets received at the target device from a source device. Out-of-order packets received at the target device are stored in the memory buffer structure in order by their sequence numbers, along with packets already in the memory buffer, if any. If a plurality of packets in the memory buffer are in order, the plurality of packets that are in order are sent to the application in the target device. The memory buffer may contain pointers that point to memory locations for storing the partially-ordered data packets, in one embodiment.

24 In yet another embodiment, the invention includes a look-ahead capability to reduce the unnecessary reordering of ROB pointers and to improve efficiency.

25 These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

26 The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

27 Fig. 1A shows, in accordance with a prior art technique, a simplified diagram showing how a source device can transmit a group of sequenced packets to a target device across a network.

28 Fig. 1B shows, in accordance with another prior art technique, a simplified diagram showing how a source device can transmit a group of sequenced packets to a target device across a network.

29 Fig. 1C illustrates the situation wherein a prior art TCB is employed to facilitate both the transmitting and the receiving processes for host computer, thereby enabling only one process to occur at any given time.

30 Fig. 2 shows, in accordance with one embodiment of the present invention, a simplified diagram showing a first host computer exchanging data with a second host computer using the inventive transmit control block (Tx TCB) and the inventive receive control block (Rx TCB) of the present invention.

31 Fig. 3 shows, in accordance with one embodiment of the present invention, a transmit control block data structure for facilitating transmitting data.

32 Fig. 4 illustrates, in accordance with one embodiment of the present invention, a transmit window,which conceptually represents the amount of data allowed to be transmitted from the transmitting host computer for a given data flow at any given time.

33 Fig. 5 illustrates, in accordance with one aspect of the present invention, a retransmit queue and a transmit pending queue.

34 Fig. 6 is a block diagram illustrating, in accordance with one embodiment of the present invention, the transmit operations involving the transmit control block of the present invention.

35 Fig. 7 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how the transmit network protocol processor may employ the transmit control block to service a request to transmit data from the host application program associated with the transmitting host computer.

36 Fig. 8 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how the transmit network protocol processor may employ the transmit control block to service a request to retransmit data from retransmit queue.

37 Fig. 9 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how the transmit network protocol processor may employ the transmit control block to send an acknowledgement to the other transmitting host computer.

38 Fig. 10 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how the transmit network protocol processor may involve the transmit control block in updating the transmit window and calculating performance data responsive to a received acknowledgement message sent from the other transmitting host computer.

39 Fig. 11 illustrates, in accordance with one embodiment of the present invention, an exemplary receive control block (Rx TCB) data structure.

40 Figs. 12A and 12B show, in accordance with embodiments of the present invention, exemplary steps for receiving data packets using the receive control block (Rx TCB).

41 Fig. 13 illustrates, in accordance with one embodiment of the present invention, the steps taken by the receive process in response to the receipt of an acknowledgement packet.

42 Fig. 14 illustrates, in accordance with one embodiment of the present invention, the steps taken by the receive process in response to the receipt of one or more data packets that contain data other than acknowledgement for data previously sent.

43 Fig. 15 illustrates, in accordance with one embodiment of the present invention, a receive transmit control block (Rx TCB) data structure for facilitating receiving data at the target device.

44 Fig. 16 illustrates, in accordance with one embodiment of the present invention, a simplified diagram showing a Rx TCB whose out-of-order packets are partially reordered.

45 Fig. 17 illustrates, in accordance with another embodiment of the present invention, a simplified diagram showing an exemplary Rx TCB that includes the partial reordering feature for out-of-order packets.

46 Fig. 18 illustrates, in accordance with another aspect of the present invention, a simplified diagram illustrating the look-ahead feature with regard to the partial reordering of out-of-order packets in a Rx TCB.

47 Appendix A, Fig. A1 is a block diagram illustrating the elements of a prior art TCP data segment;

48 Appendix A, Fig. A2a illustrates the layout in memory of a TCB data structure according to a preferred embodiment of the present invention;

49 Appendix A, Fig. A2b is a flow chart illustrating Reorder Buffer (ROB) update and processing;

50 Appendix A, Figs. A3a-A3g list the preferred transmit variables, receive variables, and re-order variables used in a plurality of algorithms needed to implement the TCP protocol;

51 Appendix A, Fig. A4 is a flow diagram illustrating the steps for transmitting a stream of data, comprising a plurality of data segments, from a sending computer to a receiving computer according to a preferred embodiment of the present invention;

52 Appendix A, Fig. A5 is a block diagram illustrating an ASIC having microcode driven processing engines for TCP/IP processing according to the present invention;

53 Appendix A, Fig. A6 illustrates how Fibre Channel frame to Gigabit Ethernet frame processing is performed in the ASIC of Appendix A, Fig. A5; and

Fig.A7 illustrates how Gigabit Ethernet frame to Fibre Channel frame processing is performed in the ASIC of Fig. A5.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

54 The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps and/or structures have not been described in detail in order to not unnecessarily obscure the present invention.

55 Fig. 2 shows, in accordance with one aspect of the present invention, a simplified diagram showing a host computer 202 exchanging data with a host computer 204 using the inventive transmit control block (Tx TCB) and the inventive receive control block (Rx TCB) of the present invention. As shown in Fig. 2, there is one transmit control block (Tx TCB) for each data flow in each of the host computers. The transmit control block (Tx TCB) is provided in a host computer and is employed during the data transmit process. Thus, each bi-directional flow of data is associated with two transmit control blocks (Tx TCBs) if both host computers at the two ends of the data flow are involved in transmitting data to one another. This is shown by transmit control blocks 206 and 208 in host computers 202 and 204 respectively. For receiving data, each host computer at the two ends of the bi-directional data flow also has a receive control block (Rx TCB). Thus, each bi-directional flow of data is associated with two receive control blocks (Rx TCBs) if both host computers at the two ends of the data flow are permitted to receive data from one another. This is shown by receive control blocks 210 and 212 in host computers 202 and 204 respectively.

56 Within each host computer, since each bi-directional data flow is provided with a separate data control block for the transmit process (i.e., the transmit control block or Tx TCB) and a separate data control block for the receive process (i.e., the receive control block or Rx TCB), that host computer can simultaneously process

transmit and receive requests using two different data control blocks. For example, host computer 202 can service simultaneously a transmit request using Tx TCB 206 and a receive request using Rx TCB 210. As the term is employed herein, processes occurring simultaneously denotes that the sub-step or sub-process of one can be executed even before the other process is finished. For example, two processes executed in two different processing circuits (such as two parallel processors) can occur simultaneously. Further, two processes executed in two different threads by a single processing circuit can be deemed to occur simultaneously even if the processing circuit, by its nature, can execute only one instruction at a time.

57 In the context of the present invention, the transmit request pertaining to certain packets pertaining to a bi-directional data flow can be serviced simultaneously with the servicing of a receive request pertaining to other packets of that bi-directional data flow. This is different from the situation in Fig. 1C wherein the host computer, such as host computer 164, must wait until the servicing of the transmit request is done and the prior art TCB (such as TCB 162) is released before having access to that TCB in order to service the receive request (or vice versa).

58 As the term is employed herein, a host computer refers to a computer-implemented device having a processor and at least one I/O port for transmitting and receiving data (hence the term "host"). It is not necessary, as the term is employed herein, for a host computer to have a keyboard or other user data entry arrangements. Further, it is not necessary, as the term is employed herein, for a host computer to have a data display device, such as a computer display screen. As such, a host computer, as the term is employed herein, may include a storage area network (SAN) arrangement, network attached storage (NAS), a data storage device having a processor for communicating with other devices via a network, and/or indeed any other computer-implemented device having a processor and capable of communicating via a network.

59 Fig. 3 shows, in accordance with one embodiment of the present invention, a transmit control block (Tx TCB) 300 data structure for facilitating transmitting data. Referring now to Fig. 3, transmit control block (Tx TCB) 300 has, among others, four major groups of transmit-facilitating parameters (i.e., parameters employed to

facilitate the transmit process): TX header, TX window management, TX queues management, and TX timer management. The more important TX header parameters include TCP source port (TCP SP) 302, TCP destination port (TCP DP) 304, IP source address (IP SA) 366, IP destination address (IP DA) 306, the sequence number to be sent next (SND_NXT) 322, the window size of the transmitting host computer (SND_WIN) 316, the Optimal Maximum Segment Size (SEG_OPT_MSS) 318, the current congestion window (ICNG_WIN/CNG_WIN) 324.

60 The more important TX window management parameters include the current congestion window (ICNG_WIN/CNG_WIN) 324, the old unacknowledged sequence number (OLD_UNA) 320, the sequence number to be sent next (SND_NXT) 322, the maximum sequence number to be sent (SND_MAX) 326, the window size of the transmitting host computer (SND_WIN) 316.

61 The more important TX queue management parameters include Transmit Pending Read Pointer (TXP_RD_PTR) 346, Transmit Pending Write Pointer (TXP_WR_PTR) 344, Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350, Retransmit Read Pointer (TXP_RTX_PTR) 348, Transmit Queue Base Address (TXP_BASE_ADDR) 330, and Transmit Queue Maximum Address (MAX_TXP_ADDR) 342.

62 The more important TX timer management parameters include the Retransmission Time Out value RTO 334, the Frame Transmit Timer FST 340, the Smoothened Round Trip Time (SRTT) 338, the Measured Round Trip Time (MRTT) 336, and the Sequence number of the segment for which timer measurements are kicked off (TMR_SEQ_NUM) 332.

63 The operation of transmit control block (Tx TCB) 300 may be better understood with reference to the figures that follow. Integral to the concept of a transmit control block (Tx TCB) are the concepts of transmit window management and queue management. Fig. 4 illustrates in a simplified format a transmit window 402, which conceptually represents the amount of data allowed to be transmitted from the transmitting host computer for a given data flow at any given time. Transmit window 402 may have a size of, for example, 64Kb although the exact value may vary

with implementations. Whenever there is a request to transmit a packet (generally from the host application software), the packet is buffered and transmit window 402 is checked (via parameter SND_WIN 316 in the transmit control block 300) to see whether there is sufficient transmit bandwidth to send out the packet.

64 Transmit window 402 may have a slow start feature; that is, transmit window 402 starts out being rather small, e.g., up to first window size 404 (generally 1 MSS or 1 Maximum Segment Size). This allows a small amount of data to be transmitted from the host computer in the beginning. If the host computer at the other end sends an acknowledgement timely, transmit window 402 gradually opens up further to allow more data to be transmitted through, up to a maximum size indicated by maximum window size 406. If an expected acknowledgement is not timely received, transmit window 402 shrinks to its minimum size again, or to some value smaller than the value it had when the acknowledgement period expires. In this manner, transmit window 402 performs the function of congestion control at the transmitting host computer.

65 Fig. 5 illustrates, in accordance with one aspect of the present invention, a retransmit queue 502 and a transmit pending queue 504. Transmit pending queue 504 is employed to queue transmit elements, which may include either the packet to be transmitted from the host computer associated with the transmit control block (Tx TCB) or, more commonly, a pointer to that packet. A transmit element may also include data associated with the packet to be transmitted such as the starting sequence number and the length of the packet.

66 If the transmit window size is sufficient to transmit a packet, that packet is transmitted and a copy (or its pointer and/or associated data) is placed into the retransmit queue 502. On the other than, a packet (or its pointer and associated data) to be transmitted is placed into transmit pending queue 504 when the transmit window size is insufficient to transmit that packet. Transmit pending queue 504 is managed by two pointers: Transmit Pending Read Pointer (TXP_RD_PTR) 346 and Transmit Pending Write Pointer (TXP_WR_PTR) 344. Transmit Pending Read Pointer (TXP_RD_PTR) 346 points to the top of the transmit queue and specifically to the queue element to be sent next when the transmit window is available. Transmit

Pending Write Pointer (TXP_WR_PTR) 344 is the tail of the transmit queue and represents the last inserted transmit element. When a packet is sent out from transmit pending queue 504, Transmit Pending Read Pointer (TXP_RD_PTR) 346 moves toward Transmit Pending Write Pointer (TXP_WR_PTR) 344. Thus, the difference between Transmit Pending Write Pointer (TXP_WR_PTR) 344 and Transmit Pending Read Pointer (TXP_RD_PTR) 346 represents the size of the transmit queue at any given point in time, which reflects the number of pending packets to be sent.

67 After being sent, a packet is moved to retransmit queue 502. In retransmit queue 502, the packet awaits acknowledgement from the other host computer before being dequeued. Retransmit queue 502 is managed by two pointers: Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 and Retransmit Read Pointer (TXP_RTX_PTR) 348.

68 In most cases, Retransmit Read Pointer (TXP_RTX_PTR) 348 and Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 point at the same retransmit queue element. During retransmit, Retransmit Read Pointer (TXP_RTX_PTR) 348 advances while Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 stays, acting as a shadow pointer. Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 is advanced only when an acknowledgement is received for the point retransmitted. If all retransmitted packets are acknowledged, Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 and Retransmit Read Pointer (TXP_RTX_PTR) 348 will catch up with one another and will point at the same queue element again.

69 For convenience, retransmit queue 502 and transmit pending queue 504 may be implemented together as a circular queue and occupy a contiguous block of memory. However, the use of Retransmit Read Pointer (TXP_RTX_PTR) 348, Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350, Transmit Pending Read Pointer (TXP_RD_PTR) 346 , and Transmit Pending Write Pointer (TXP_WR_PTR) 344 allow this retransmit queue 502 and transmit pending queue 504 to be managed as two different logical queues.

70 Fig. 6 is a block diagram illustrating, in accordance with one embodiment of the present invention, the transmit operations involving the transmit control block (Tx

TCB) of the present invention. Referring to Fig. 6, there is shown a transmit network protocol processor (TX NPP) 602, representing the processor servicing the transmit operations. In the example of Fig. 6, transmit network protocol processor (TX NPP) 602 represents a micro-code driven engine, or a plurality of micro-code driven engines, which takes in, among others, four different types of requests: Request to send data, request to retransmit, request to send acknowledgement (ACK), and request to update transmit window based on acknowledgement received. In general, these four requests may be serviced concurrently by transmit network protocol processor (TX NPP) 602 using four independently executing threads.

71 The Requests to send data are typically generated by the application software associated with the transmitting host computer. These requests are queued up in a queue 604 while awaiting to be serviced by transmit network protocol processor (TX NPP) 602. In one embodiment, the content of each queue element of queue 604 includes the pointer to the transmit control block (Tx TCB) associated with the data flow, the pointer to the buffer element where the data is stored, and the length of the data to be transmitted.

72 Since there may be multiple data flows between a given host computer and another host computer or other host computers, each host computers may have multiple transmit control blocks (Tx TCBs), each associated with a data flow. The plurality of transmit control blocks (Tx TCBs) are shown in Fig. 6 by reference number 612. The transmit control block (Tx TCB) for a given data flow may be found using, for example, the TCP source port (TCP SP), the TCP destination port (TCP DP), the IP destination address (IP DA), and/or the IP source address (IP SA).

73 The requests to retransmit are typically generated when a retransmit timer expires after some predefined time has passed since a given packet was transmitted, and an acknowledgement has not yet been received. Since there may be multiple data flows between a given host computer and another host computer or other host computers, each host computers may have multiple retransmit timers. The plurality of retransmit timers are shown in Fig. 6 by reference number 614. These requests are queued up in a queue 606 while awaiting to be serviced by transmit network protocol processor (TX NPP) 602. In one embodiment, the content of each queue element of

queue 606 includes the pointer to the transmit control block (Tx TCB) associated with the data flow.

74 The requests to send acknowledgement (ACK) are generated responsive to the successful receipt of data transmitted from the other host computer. These requests are queued up in a queue 608 while awaiting to be serviced by transmit network protocol processor (TX NPP) 602. In one embodiment, the content of each queue element of queue 608 includes the pointer to the transmit control block (Tx TCB) associated with the data flow, the acknowledgement number associated with the data received, and the window size associated with the data received.

75 The requests request to update transmit window based on acknowledgement received are generated responsive to the acknowledgement received . These requests are queued up in a queue 610 while awaiting to be serviced by transmit network protocol processor (TX NPP) 602. In one embodiment, the content of each queue element of queue 610 includes the receive window size, the acknowledge number received, the pointer to the Tx TCB, and optionally time stamp and SACK information.

76 In one embodiment, these queues 604, 606, 608, and 610 are serviced by transmit network protocol processor (TX NPP) 602 in a round-robin manner although any other scheme to ensure that the requests in these queues are serviced in an efficient manner may also be employed.

77 The operation of the transmit network protocol processor (TX NPP) 602 and the role of the transmit control block (Tx TCB) in servicing the requests associated with the transmitting operations may be better understood with reference to the flowcharts of Figs. 7, 8, 9, and 10 herein. In one embodiment, the processes illustrated in Figs. 7-10 represent processes offloaded from the host processor of the host computer in order to improve data throughput. By way of example, these processes may be implemented by circuitry comprising one or more network processors and/or embedded processors and/or co-processors operating in parallel with the host processor of the host computer. The aforementioned circuitry may be implemented in an integrated manner with the host computer or may be implemented

on a plug-in card, such as a network interface card (NIC), that is configured to be detachably coupled to the bus of the host processor.

78 Fig. 7 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how transmit network protocol processor (TX NPP) 602 may employ the transmit control block (Tx TCB) to service a request to transmit data from the host application program associated with the transmitting host computer. Once the application requests to send out data, that data is fetched and placed into buffer memory. Each request is associated with a particular data flow, and hence a particular transmit control block (Tx TCB). Using the TCP source port (TCP SP), the TCP destination port (TCP DP), the IP destination address (IP DA) and/or the IP source address (IP SA), the transmit control block (Tx TCB) associated with the request is then obtained. The request is then placed into queue 604 to be serviced by transmit network protocol processor (TX NPP) 602 as discussed earlier.

79 In block 702, the method first check to see whether the transmit window is available to send the requested data. With reference to Fig. 3, the current window size is obtained from the transmit control block (Tx TCB), and more particularly, from the parameter SND_WIN 316 of the associated transmit control block (Tx TCB). If the window is available to transmit the requested data packet, the method performs certain housekeeping functions prior to sending the requested data packets onward to be sent out. Thus, the method proceeds to block 704 to update the transmit window size of the transmitting host computer (SND_WIN) 316, i.e., to reduce the size of the transmit window as part of transmit window management. Further, the method may also update the sequence number to be sent next (SND_NXT) 322 so that that the sequence number to be sent next (SND_NXT) 322 can reflect the next sequence number to be used for transmitting the next packet. Additionally, the method may also update TMR_SEQ_NUM 332, thereby starting a timer so that the delay can be measured once the acknowledgement is received.

80 In block 706, the method also places the transmitted packet(s) into retransmit queue 502, where they will stay until an acknowledgement is received. Thus, the Transmit Pending Read Pointer (TXP_RD_PTR) 346 is advanced. Furthermore Frame Transmit Timer FST 340 is also updated.

81 In block 708, the retransmit timer 614 is started for this data flow if it has not been started earlier. In block 710, the request, along with the transmit control block (Tx TCB) pointer, is send from transmit network protocol processor (TX NPP) 602 to Header Preparation Network Protocol Processor (HDR NPP) 616 (see Fig. 6) to prepare the header along with the data to send to the receiving host computer.

82 In one embodiment, the following information is transmitted from transmit network protocol processor (TX NPP) 602 to header preparation processor (HDR NPP) 616 to facilitate the preparation of a data packet (including the packet header) for transmission: pointer to the buffer where the data resides, transmit control block (Tx TCB) pointer, sequence #, length, acknowledge #, window size, and flags such as PSH, ACK, or URG (which are used in the TCP header). Once transmit network protocol processor (TX NPP) 602 sends this information to header preparation processor (HDR NPP) 616, the process for servicing the current transmission request ends.

83 On the other hand, if the transmit window is not available (as determined in block 702), the method proceeds from block 702 to block 714 to queue the packet on the transmit pending queue. Thus Transmit Pending Write Pointer (TXP_WR_PTR) 344 is updated to reflect the fact that the packet has been queued at the tail of the pending transmit queue.

84 Fig. 8 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how transmit network protocol processor (TX NPP) 602 may employ the transmit control block (Tx TCB) to service a request to retransmit data from retransmit queue 502. The process of Fig. 8 is invoked when the retransmit timer RTX TMR 614 associated with this data flow has expired. In this case, the process will send out one or more MSS-size packets (maximum segment size) from the retransmit queue, the exact number of MSS to be sent depends on ICNG_WND (324), which tracks congestion.

85 In block 804, the packets to be retransmitted (the exact number of which is determined in block 802) are sent out. As far as the management of retransmit queue 502 is concerned, this involves keeping Retransmit Shadow Pointer

(TXP_SHD_RD_PTR) 350 at the head of retransmit queue 502 and moving Retransmit Read Pointer (TXP_RTX_PTR) 348 to reflect the number of packets sent out.

86 In block 806, a set of data is acquired from both the retransmit queue element(s) associated with the packet(s) to be retransmitted and the associated transmit control block (Tx TCB). For example, data such as the buffer pointer, the sequence number, the length of the data to be retransmitted may be obtained from the retransmit queue element. From the associated transmit control block (Tx TCB), the data obtained may include the acknowledgement number, the receive window size (354 of Fig. 3). Furthermore, since a retransmit is about to be performed, the process adjusts the current congestion window (ICNG_WIN/CNG_WIN) 324. As mentioned earlier, the transmit window is reduced after data is transmitted or retransmitted until acknowledgement is received.

87 In block 808, the retransmit request along with the transmit control block (Tx TCB) pointer is send from transmit network protocol processor (TX NPP) 602 to header preparation processor (HDR NPP) 616 (see Fig. 6) to prepare the header along with the retransmit data to send to the receiving host computer. In one embodiment, the following information is transmitted from transmit network protocol processor (TX NPP) 602 to header preparation processor (HDR NPP) 616 to facilitate the preparation of a data packet (including the packet header) for transmission: pointer to the buffer where the data resides, transmit control block (Tx TCB) pointer, sequence #, length, acknowledge #, window size, and TCP-related flags such as PSH, ACK, or URG. Once transmit network protocol processor (TX NPP) 602 sends this information to header preparation processor (HDR NPP) 616, the process for servicing the current retransmit request of Fig. 8 ends.

88 Fig. 9 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how transmit network protocol processor (TX NPP) 602 may employ the transmit control block (Tx TCB) to send an acknowledgement to the other transmitting host computer. The process of Fig. 8 is invoked after the host computer has successfully received data packet(s) from the other transmitting host computer. In block 902, it is recognized that one or more data

packet(s) has been successfully received at the host computer. Based on the received data sequence, the Rx process prepares the ACK # to be sent and the receive window size and queues the request to TX NPP to send the ACK. Furthermore, the information in the received data packet(s) is also employed to obtain a pointer to the transmit control block (Tx TCB) associated with this data flow. In block 904, the current sequence number is obtained from the sequence number to be sent next (SND_NXT) 322.

89 In block 906, the information obtained in blocks 902 and 904 is sent to header preparation processor (HDR NPP) 616 to prepare a special acknowledgement packet to be sent to the other transmitting host computer. In one embodiment, the following information is transmitted from transmit network protocol processor (TX NPP) 602 to header preparation processor (HDR NPP) 616 to facilitate the preparation of an acknowledgement packet (including the packet header) for transmission: transmit control block (Tx TCB) pointer, sequence #, length, acknowledge #, window size. Once transmit network protocol processor (TX NPP) 602 sends this information to header preparation processor (HDR NPP) 616, the process for sending an acknowledgement of Fig. 9 ends.

90 Fig. 10 is a simplified flowchart illustrating, in accordance with one embodiment of the present invention, how transmit network protocol processor (TX NPP) 602 may involve the transmit control block (Tx TCB) in updating the transmit window and calculating performance data responsive to a received acknowledgement message sent from the other transmitting host computer. Generally speaking, there are two main tasks that need to be handled: (1) updating the parameters that manage the transmit window and, as a result, sending out data packets if the updated window size allows such transmitting, and (2) calculating performance data based on the information in the transmit control block (Tx TCB) and the information received in the acknowledgement message.

91 In block 1002, the TX window parameters are updated. As can be seen, the parameters updated are the window size of the transmitting host computer (SND_WIN) 316, the old unacknowledged sequence number (OLD_UNA) 320, the Frame Transmit Timer FST 340, and TMR_SEQ_NUM 332. In block 1004,

performance data is calculated and the following parameters are updated: Retransmission Time Out value RTO 334, Smoothened Round Trip Time (SRTT) 338, Measured Round Trip Time (MRTT) 336.

92 In block 1006, the process releases any data packet(s) held for retransmit (as a result of an earlier transmit from either transmit pending queue 504 or retransmit queue 502) if such data packet(s) pertain to the received acknowledgement message. Thus, the process updates Retransmit Shadow Pointer (TXP_SHD_RD_PTR) 350 and/or Retransmit Read Pointer (TXP_RTX_PTR) 348 as appropriate. In block 1008, the process determines whether there is any pending data packets in transmit pending queue 504 to be sent out.

93 In one embodiment, if the difference between the head of transmit pending queue 504 (i.e., Transmit Pending Read Pointer (TXP_RD_PTR) 346) and the tail of transmit pending queue 504 (i.e., Transmit Pending Write Pointer (TXP_WR_PTR) 344) is greater than zero, the process deems there is pending data packets awaiting transmittal. In this case, the process outputs a request to cause the transmit network protocol processor (TX NPP) to invoke the process of Fig. 7 to perform data transmittal. This is seen in block 1010. On the other hand, if there is no more pending data packets in transmit pending queue 504 (as determined in block 1008), the process of Fig. 10 ends at block 1012.

94 As mentioned earlier, the use of a separate transmit control block (Tx TCB) and a separate receive control block (Rx TCB) to handle the transmitting and receiving tasks simultaneously at a given host computer for a given data flow eliminates the bandwidth bottleneck found in the prior art when a transmit control block is employed for both the transmitting and receiving tasks for the data flow. Fig. 11 illustrates, in accordance with one embodiment of the present invention, an exemplary receive control block (Rx TCB) data structure 1100. Rx TCB 1100 is employed to store receive-facilitating parameters, i.e., parameters employed to facilitate the receive process. Receive control block (Rx TCB) 1100 has, among others, parameters to enable a plurality of functions, including: reordering out-of-order packets, allowing the receive network protocol processor (RCV NPP), instead of the

transmit network protocol processor (TX NPP), to send out an acknowledgement packet, and maintaining Rx window size on a connection-by-connection basis.

95 With respect to out-of-order packet reordering, there is provided, in accordance with one aspect of the present invention, a method for partially reordering out-of-order packets so as to render the packet receiving process more bandwidth efficient. Suppose the receive network protocol processor (RCV NPP) was expecting packets having sequence numbers 1 to 1000. However, before the packets having sequence numbers 1-1000 are received, the receive network protocol processor (RCV NPP) receives instead packets having sequence numbers 1002 to 64,000. In some prior art implementations, these out-of-order packets would be discarded, requiring the transmitting host computer to send them again after the expected packets (i.e., those having sequence numbers 1-1000) are received.

96 In accordance with one aspect of the present invention, the receive network protocol processor (RCV NPP) would keep the packets having sequence numbers 1002 to 64,000, partially reordering them as necessary, but store them in the out-of-order buffers instead of sending them to the host software within the receiving host computer. The partially ordered packets may be stored, for example, in a linked list using the sequence number of the packets as the key.

97 By partially reordering the out-of-order packets (the re-ordering is only partial since packets having sequence numbers 1-1000 are still missing), the packets having sequence numbers 1-64,000 can be very quickly assembled once the packets having sequence numbers 1-1000 arrive. This rapid assembly permits the receive network protocol processor (RCV NPP) to more quickly send the complete data to the higher layer software. Furthermore, by reducing the amount of retransmission that the transmitting host computer has to perform, the invention effectively increases the bandwidth of the communication channel between the two host computers. This aspect of the invention is discussed in greater detail in a co-pending application entitled "Methods And Apparatus For Handling Out-Of-Order Packets," filed on even date and incorporated by reference (Attorney Docket No. ATECP002-R4/SNG-028A).

98 In receive control block (Rx TCB) 1100, there is shown a plurality of reorder buffers (ROBs): ROB1, ROB2, ROB3, and ROB4. There may be as many reorder buffers as desired although only four are shown. Each ROB is employed to store parameters associated with an out-of-order packet. The parameters associated with each ROB includes the following: Pointer to the data (PKT_PTR), Storage Transport Layer Header (STL, which is employed for Fiber Channel Frame Reassembly), Length of current packet (PKT_LEN), and Sequence number of current packet (SEQ_NUM).

99 With reference to ROB1, for example, these four parameters are shown by reference numbers 1104, 1106, 1108, and 1110 respectively. To provide for storage flexibility and to avoid making each receive control block (Rx TCB) unduly large, the out-of-order buffers may be extended into the memory space of the host computer system. Thus, if a greater number of reorder buffers are required beyond what is provided in the receive control block (Rx TCB) data structure, a pointer (ROB EXTENSION PTR) 1112 is provided, which points to a location in the memory space of the host computer system where the extension reorder buffers may be found.

100 The role of the receive control block (Rx TCB) in the data receiving operations may be better understood with reference to Figs. 12A-14 that follow. In one embodiment, the processes illustrated in Figs. 12A-14 represent processes offloaded from the host processor of the host computer in order to improve data throughput. By way of example, these processes may be implemented by circuitry comprising one or more network processors and/or embedded processors and/or co-processors operating in parallel with the host processor of the host computer. The aforementioned circuitry may be implemented in an integrated manner with the host computer or may be implemented on a plug-in card, such as the aforementioned network interface card (NIC), that is configured to be detachably coupled to the bus of the host processor.

101 Fig. 12A shows, in accordance with one embodiment of the present invention, exemplary steps for receiving data packets using the receive control block (Rx TCB). In block 1202, the sequence number(s) of the received data packet(s) are compared with the expected sequence number(s), which are stored by RCV_NXT 1120 in the receive control block (Rx TCB). The receive control block (Rx TCB) for the received packet may be found using information such the TCP source port, the TCP destination

port, the IP source address, and/or the IP destination address. If the two match, then the received packet(s) are deemed in order and the method proceeds to block 1204. On the other hand, if there is a discrepancy, the received packet(s) are deemed out of order, and the method proceeds to block 1248, which is described further in Fig. 12B herein.

102 In block 1204, the method updates RCV_NXT 1120, which updates the sequence number expected for the next packet(s) received. Furthermore, the receive window is updated by updating RCV_WIN 1122, which reduces the receive window size temporarily until data is sent from the receive network protocol processor (RCV NPP) to the application software in the receiving host computer.

103 Once updating is performed in block 1204, the method proceeds to block 1206 wherein the method decides whether the received packet pertains to received acknowledgement for data sent earlier from this host computer, or whether the received data packet pertains to data, other than an acknowledgement, sent from the other transmitting host computer. If the received data packet pertains to an acknowledgement associated with data sent earlier from this host computer, the method proceeds to block 1208 wherein the received acknowledgement is sent to the transmit network protocol processor (TX NPP) so that the transmit network protocol processor (TX NPP) can update its window, calculate performance data, and the like. This aspect has been discussed earlier in connection with Fig. 10 herein.

104 On the other hand, if the received data packet pertains to data, other than an acknowledgement, sent from the other transmitting host computer, the method proceeds to block 1210 wherein the data is moved into the host computer's memory, e.g., into the host buffer memory space, for use by the host software application. In blocks 1212 and 1214, the acknowledgement procedure is undertaken. In one embodiment, a cumulative acknowledgement scheme is employed (but not required in every implementation). Under the cumulative acknowledgement scheme, an acknowledgement is sent out for every other packet received to optimize bandwidth. Thus, in block 1212, a delayed acknowledgement timer is started if it has not been started already.

105 On the other hand, in block 1214, if the delayed acknowledgement timer has already started, the method stops the timer and proceeds to send out an acknowledgement to the other transmitting host computer. In any event, if the delayed acknowledgement timer expires, an acknowledgement is sent out to the transmitting host computer, in which case the acknowledgement only pertains to the one (instead of two) packets received. Generally speaking, the delayed acknowledgement timer may be set to a given value, e.g., 100ms. When the delayed acknowledgment timer expires, a request is then queued in the Rx NPP queue.

106 In block 1216, the method sends the data necessary to create an acknowledgement packet to the transmit network protocol processor (TX NPP) of the host computer to send out an acknowledgement. This data includes, for example, the acknowledgement number, the window size, and the pointer to the transmit control block (Tx TCB). This procedure has been described earlier in connection with Fig. 9 herein. The Rx NPP also has the option to send out an acknowledgement by itself. For example, the Rx NPP may send an immediate acknowledgement if it receives out-of-order data, duplicate data, or Rx window probe.

107 In block 1218, the method checks to determine whether the newly received data packets made the earlier partially reordered packets, which are stored in the reordered buffers, in order. If not, the process returns in block 1218. On the other hand, if the receipt of the newly received data packets made the earlier partially reordered packets in order, the method proceeds to block 1204 again to send out all the ordered packets to the host application buffer space and to send out acknowledgement for the same.

108 Fig. 12B represents the situation in Fig. 12A wherein the received packet is found to be out of order (as determined by block 1202 of Fig. 12A). In block 1250, the received packet is examined to determine whether it is an acknowledgement from the other transmitting host computer for data previously sent by this host computer or it is data, other than an acknowledgement, sent by the other transmitting host computer. If the received packet is found to be an acknowledgement, the method proceeds to step 1208 of Fig. 12A, i.e., the acknowledgement data is sent to the transmit network protocol processor (TX NPP) in order to allow the transmit network

protocol processor (TX NPP) to, among other tasks, update its window, calculate performance data, and the like.

109 On the other hand, if the received packet is found to be out of order data, other than an acknowledgement, sent from the other transmitting host computer, the method proceeds to block 1252 to determine whether the received data fits within the receive window. The receive window may be currently narrow because the host computer received a chunk of data earlier and has not completely moved the data received earlier into the application buffer memory, and thus can accept only a limited amount of new data. If the data does not fit within the received window, the method proceeds to block 1254 wherein the received data is simply discarded. Thereafter, the method returns to wait for the arrival of new data. This is shown in block 1256.

110 On the other hand, if the received data fits within the window, the method proceeds to block 1216 of Fig. 12A, i.e., it sends the received data, along with the receive control block (Rx TCB) to have the acknowledgement packet prepared and sent out by the transmit network protocol processor (TX NPP). The acknowledgement, in this case, will indicate to the other transmitting computer that the packets are received but they are out of order.

111 Each ROB is employed to store parameters associated with an out-of-order packet. The parameters associated with each ROB includes the following: Pointer to the data (PKT_PTR 1104), Storage Transport Layer Header (STL 1106), Length of current packet (PKT_LEN 1108), and Sequence number of current packet (SEQ_NUM 1110).

112 Further, the received data is stored in the out-of-order buffer as shown in block 1258. As discussed earlier, this out-of-order buffer is managed by parameters such as PKT_PTR 1104, STL 1106, PKT_LEN 1108, and SEQ_NUM 1110 in a linked list of reorder buffers, which is maintained in the receive control block (Rx TCB) and extends into the memory of the host computer as needed via the use of the buffer extension pointer ROB EXTENSION PTR 1112. Thus, in block 1260, the parameters associated with the reorder buffers in the receive control block (Rx TCB) are updated. Note that the Storage Transport Layer Header (STL) parameter

associated with the reorder buffers is modified only if the protocol (indicated by PROTOCOL 1124 in the receive control block (Rx TCB)) indicates that the current protocol is Fiber Channel over IP (FC/IP) over TCP/IP. Thereafter, the method returns to wait for the arrival of new data. This is shown in block 1262.

113 The separation of the transmit network protocol processor (TX NPP) from the receive network protocol processor (RCV NPP), in addition to the separation of the transmit control block (Tx TCB) from the receive control block (Rx TCB), advantageously allows the host computer to communicate at a higher transmission bandwidth. As mentioned above, each of the transmit network protocol processor (TX NPP) and receive network protocol processor (RCV NPP) executes their own threads for handling a variety of requests. Between the transmit network protocol processor (TX NPP) and the receive network protocol processor (RCV NPP), there is a substantial amount of required coordination.

114 In accordance with one aspect of the present invention, the receive network protocol processor (RCV NPP) passes data onto the transmit network protocol processor (TX NPP) on at least two occasions to update the transmit control block (Tx TCB) and other parameters associated with the transmit process. These two occasions include the receipt of an acknowledgement from the other transmitting host computer, and the receipt of data sequence information, which allows the transmit network protocol processor (TX NPP) to send an acknowledgement to the other transmitting host computer.

115 In the exemplary embodiment illustrated by Fig. 13, the receipt of one or more acknowledgement packets from the other transmitting host computer causes the receive network protocol processor (RCV NPP) 1302 to queue data associated with the received acknowledgement packet(s) into a queue 1304. Transmit network protocol processor (TX NPP) 1306 takes queue elements from queue 1304 in order to update its transmit control block (Tx TCB), the window size, calculate performance data, update the retranmsit buffer, and the lik. The TX NPP may then update the RX window based on the received acknowledgement. Furthermore, it may also update the send ACK # and the window size.

116 Among the information sent by receive network protocol processor (RCV NPP) 1302 to transmit network protocol processor (TX NPP) 1306 are the acknowledgement number, the time stamp value reflecting the time the packet(s) for which the acknowledgement is sent is received by the other transmitting host computer (the time stamp value is employed by the transmit network protocol processor (TX NPP) to calculate the delay based on the current time value), information associated with selective acknowledgement blocks such as the start sequence number (START_SEQ) and the associated length, the window size information from the other host transmitting computer, and the pointer to the transmit control block (Tx TCB).

117 In the exemplary embodiment illustrated by Fig. 14, the receive network protocol processor (RCV NPP) sends information to the transmit network protocol processor (TX NPP) to send out an acknowledgement . This may occur because the delayed acknowledgement timer has expired or because two packets back-to-back has been received and an acknowledgement should be sent. The receive network protocol processor (RCV NPP) 1402 queues requests to send acknowledgement into a queue 1404. Transmit network protocol processor (TX NPP) 1406 takes queue elements from queue 1404 in order to computer parameters necessary to send out an acknowledgement and also to update the transmit control block (Tx TCB) (e.g., thee latest ACK and window size sent).

118 Among the information sent by receive network protocol processor (RCV NPP) 1402 to transmit network protocol processor (TX NPP) 1406 are the sequence number and the length, which allows the transmit network protocol processor (TX NPP) to calculate the acknowledgement number. Other information includes the window size to be sent to the other transmitting host computer, and the pointer to the transmit control block (Tx TCB). This information taken off the top of queue 1404 will be forwarded by transmit network protocol processor (TX NPP) 1406 to the header preparation processor (HDR NPP) create an acknowledgement packet for sending to the other host transmitting computer.

119 With respect to partially reordering packets, the invention involves techniques and structure for partially reordering packets in both a TCB that handles both transmitting

and receiving as well as in a control block arrangement in which there is a separate transmit TCB and a separate receive TCB (as discussed above herein). It should be noted that although the discussion below pertains to one implementation of partially reordering packet in a TCB that handles both transmitting and receiving, embodiments of the invention herein also applies to a control block arrangement in which there is a separate transmit TCB (Tx TCB) and a separate receive TCB (Rx TCB) for each networked device.

120 Fig. 15 illustrates, in accordance with one embodiment of the present invention, a receive transmit control block (Rx TCB) 1502 data structure for facilitating receiving data at the target device. Together with the transmit TCB (Tx TCB), Rx TCB 1502 facilitates the receive process at a target device by managing various aspects of the data transfer, such as the numbers of packets that are in transit at any given time, also called window management, and also by managing packet queuing and timing.

121 Incoming packets, themselves, are not directly kept in the re-order buffer (ROB) 1504 of the Rx TCB 1502. Since packets can be of varying size, it may be more efficient to store ROB pointers ("pkt_ptrs") 1506 to the packets in the ROB 1504, and actually store some or all the packets themselves in an external memory location. Along with each pointer 1506, the packet length ("PKT_LEN[10:0]") 1508 and the sequence number ("SEQ_NUM") 1510 are stored. Of course the size of the packet length 1508 may be varied according to implementations.

122 The Rx TCB 1502 also contains a ROB extension pointer ("ROB extension ptr") 1512 for packet overflow. That is, arriving packets whose ROB pointers 1506 cannot fit in the ROB 1504 are themselves stored in a memory location that is referenced by the ROB extension pointer 1512. It is, in essence, a pointer to a location that contains other pointers, which can in turn reference packets.

123 The Rx TCB stores ROB pointers in its ROB 1504 in sequence number order. For example, the ROB pointer with the smallest sequence number is stored at the top 1516 of the ROB 1504, and the ROB pointer with the largest sequence number is stored at the bottom 1522 of the ROB 1504, or is referenced in a memory location by the ROB extension pointer 1512. ROB pointers in this memory location may also be stored by

sequence number. As new packets arrive, the ROB is reordered as necessary to insure that the ROB pointers are kept in order by sequence number. Also, packets may be discarded when their ROB pointers can no longer fit into the ROB 1504, or with the ROB extension pointer 1512, after readjustment.

124 Fig. 16 illustrates, in accordance with one aspect of the present invention, a simplified diagram showing an Rx TCB whose out-of-order packets are partially reordered. The reordering is only partial since reordering is undertaken even if all of the packets being reordered are not sequentially numbered. For example, reordering packets occurs in the buffer even though their numbers are not contiguous. The TCB in the example of Fig. 16 contains three slots in the ROB 1624 for each of three ordered entries, a first entry 1604, and second entry 1606, and a third entry 1608. Of course the number of slots can be varied as desired. Each entry may include a pointer to the received packet stored in the ROB, the length of the packet, and the sequence number of the packet. The TCB may also contain a counter 1601 that keeps track of the number of out-of-order packets that have arrived at the target device.

125 Once the packet with the expected sequence number 1602 arrives at the target device, the Rx TCB forwards it, along with all sequentially contiguous packets found in the ROB, to the application. The TCB determines the last sequentially contiguous entry in the memory buffer 1624 by subtracting the out-of-order counter 1601 from the total amount of packets received and stored.

126 Suppose three packets are transmitted from the source device to the target device. The first transmitted packet has a sequence number of 100, the second transmitted packet has a sequence number of 1000, and the third transmitted packet has a sequence number of 2000.

127 The packets arrive, however, out of order. The first packet received is the third transmitted packet with a sequence number of 2000. The second packet received is still the second transmitted packet with a sequence number of 1000. And the third packet received is the first transmitted packet with a sequence number of 100.

128 Initially, at time =N (322), prior to the arrival of any packets, the ROB 1624 is empty. The expected sequence number 1602 is 100 which is also the sequence

number of the first packet to be sent from the source device, since the TCP/IP data transfer has not yet started.

129 At time =N +1 (310), the third transmitted packet arrives with sequence number 2000. Since this does not match the expected sequence number 1602 of "100", the third transmitted packet is placed in memory, and a ROB pointer 1616 is stored in the ROB 1624 at the first entry 1604, along with its length and sequence number. The out-of-order counter 1601 changed to "1". That is, there is one entry, and it is not in proper order.

130 At time =N +2 (312), the second transmitted packet has arrived with sequence number 1000. Since this also does not match the expected sequence number 100, the second transmitted packet is placed in the memory, and a ROB pointer 1618 is stored in the ROB 1624, along with its length and the sequence number. Since the second transmitted packet has a sequence number smaller than the previously arrived third transmitted packet, the ROB pointer entries in the ROB 1624 are reordered. The ROB pointer to the second transmitted packet is placed in the first entry 1604, while the ROB pointer to the third transmitted packet is moved to the second entry 1606. The out-of-order counter 1601 is now changed to "2". That is, there are two entries, and they are both out of order, albeit partially reordered in ROB 1624.

131 At time =N +3 (314), the first transmitted packet finally arrives, with sequence number that matches the expected sequence number 1602 of "100". At this point, the TCB places the first transmitted packet in memory, stores a ROB pointer 1620 in the ROB 1624 at the first entry 1606, along with its length and the sequence number. And again, the ROB 1624 is reordered. The ROB pointer to the second transmitted packet is moved to the second entry 1606, and the ROB pointer to the third transmitted packet is moved again to the third entry 1608. The out-of-order counter 1601 is now "0", since there are three entries, but all are corrected ordered.

132 The TCB then forwards all three properly ordered packets by sequence number to the application on the target device, clears the ROB, and subsequently determines the next expected sequence number 1602.

133 In the context of the present invention, providing a ROB for arrived packets reduces the need to flood the network with retransmitted packets. Also, reordering the ROB pointers by sequence number eliminates the need to continuously re-inspect the sequence number of each packet, thereby reducing network latency for the application.

134 Fig. 17 illustrates, in accordance with another embodiment of the present invention, a simplified diagram showing an exemplary Rx TCB that includes the partial reordering feature for out-of-order packets. In the embodiment of Fig. 17, packets that are sequentially contiguous with the packet having the next expected sequence number are forwarded to the application in the target device, and packets that are not sequentially contiguous with those forwarded are kept in the buffer to be reordered with packets subsequently arrived. In the example of Fig. 17, the Rx TCB contains three slots in the ROB 1724 for each of three ordered entries, a first entry 1704, and second entry 1706, and a third entry 1708. Each entry comprises a pointer to the received packet stored in the ROB, the length of the packet, and the sequence number of the packet. As in the Fig. 16, the TCB also contains a counter 1701 that keeps track of the number of out-of-order packets that have arrived at the target device.

135 In this example, there are four packets to be transmitted from the source device to the target device. The first transmitted packet has a sequence number of 100, the second transmitted packet has a sequence number of 1000, and the third transmitted packet has a sequence number of 2000, and the fourth transmitted packet has a sequence number of 3000.

136 The packets arrive, however, out of order. The first packet received is the fourth transmitted packet with a sequence number of 3000. The second packet received is still the second transmitted packet with a sequence number of 1000. And the third packet received is the first transmitted packet with a sequence number of 100. The third transmitted packet has not arrived.

137 Initially, at time =N (422), prior to any packets arriving, the ROB 1724 is empty. The expected sequence number 1702 is 100 which is also the sequence number of the

first packet to be sent from the source device, since the TCP/IP data transfer has not yet started.

138 At time =N +1 (410), the fourth transmitted packet arrives with sequence number 3000. Since this does not match the expected sequence number 100, the fourth transmitted packet is placed memory, and a ROB pointer 1716 is stored in the memory buffer 1724 at the first entry 1704, along with its length and sequence number. The out-of-order counter 1701 is changed to "1". That is, there is one entry, and it is not in proper order.

139 At time =N +2 (412), the second transmitted packet has arrived with sequence number 1000. Since this also does not match the expected sequence number 100, the second transmitted packet is placed memory, and a ROB pointer 1718 is stored in the memory buffer 1724, along with its length and the sequence number. Since the second transmitted packet has a sequence number smaller than the previously arrived fourth transmitted packet, the ROB pointer to the second transmitted packet is placed in the first entry 1704, and the ROB pointer to the fourth transmitted packet is moved to the second entry 1706. The out-of-order counter 1701 is changed to "2", since there are two entries, and both are out of order.

140 At time =N +4 (414), the first transmitted packet finally arrives, with a sequence number that matches the expected sequence number 1702. At this point, the TCB places the first transmitted packet in memory, stores a ROB pointer 1720 in the ROB 1724 at the first entry 1706, along with its length and the sequence number. The ROB pointer to the second transmitted packet is moved to the second entry 1706, and the ROB pointer to the fourth transmitted packet is moved to the third entry 1708. The out-of-order counter 1701 is now changed back to "1", since there are three total entries, the first two are in proper order, but the last one is out-of-order.

141 The TCB then forwards the first two properly ordered entries, the first & second transmitted packets, by sequence number to the application on the target device. It then moves ROB pointer to the fourth transmitted packet to the first entry 1704. And the TCB subsequently determines the next expected sequence number 1702.

142 In the context of the present invention, partially reordering the ROB pointers, and forwarding the group of packets by sequence number to the application without continuous re-inspection, reduces network latency for the application, as in the prior art shown in Fig. 15

143 Fig. 18 illustrates, in accordance with another aspect of the present invention, a simplified diagram showing the look-ahead algorithm with regard to the partial reordering of out-of-order packets in a TCB 1802. The TCB displays a ROB 1824 that can hold a total of five ROB pointer entries.

144 The source device has sent six packets, with sequence numbers of 100, 1000, 2000, 3000, 4000, and 5000 respectively. The target device has received just three of the packets, with sequence numbers 2000, 3000, and 4000 respectively. The corresponding ROB pointers are stored by sequence number in the ROB 1824.

145 The TCB look-ahead algorithm predicts, based on the lengths and sequence numbers of the already arrived packets, the next expected sequence number 1808, and/or the maximum transport unit (MTU) the number of packets that have not arrived and allocate space therefor in the buffer. The MTU is typically determined during, for example, the initialization process (e.g., during discovery). Suppose, for example, that the MTU is 1,500 bytes. In this case, the look-ahead algorithm would predict that at least two packets will be required to transport the first 1,900 bytes (i.e., the smallest received sequence number of 2000 minus the expected sequence number of 100).

146 Accordingly, the look-ahead algorithm allocate the next two entries in the ROB for the expected two packets. Any packet arriving whose sequence number is not between the smallest received sequence number of 2,000 and the expected sequence number of 100 is simply discarded. For example, if the packet with sequence number 5,000 arrives next, it will be discarded because the remaining two entries have already been allocated. On the other hand, if the packet with the expected sequence number 100 or a packet with a sequence number 1,000 arrives next, that packet will be put into the ROB and be taken into consideration during re-ordering.

147 In the context of the present invention, the TCB look-ahead algorithm reduces the unnecessary reordering of ROB pointers. If the TCB believes that that a packet with a higher sequence number needs to be discarded in the future to make room for other expected packets, it will discard that packet upon its arrival at the target device. This thereby reduces the overall network latency for the application, as groups of correctly ordered packets can be forwarded to the application efficiently by reducing the amount of reordering steps that need to occur.

Appendix A: Another Embodiment of the Invention

148 This invention relates generally to protocols for sending and receiving data across one or more computer networks and more specifically to implementing in hardware the Transmission Control Protocol/Internet Protocol used to transport data between two computers.

149 The Transmission Control Protocol/Internet Protocol ("TCP/IP") is a combination of protocols which allows computers (or "hosts") to communicate information across one or more networks. Within TCP/IP, each protocol is assigned a specific task, and the protocols are arranged in a layered design typically referred to as a "stack", i.e., a higher level protocol uses a lower level protocol to perform its tasks. Between TCP and IP, TCP is the higher level protocol. Moreover, TCP/IP processing is typically performed in software. Although TCP and IP are two of a family of many different protocols which perform their own unique tasks, it has become traditional to refer to the whole family of protocols as TCP/IP simply because TCP and IP are the best known of these protocols.

150 The Internet Protocol ("IP") is the protocol that provides for an unreliable transmission of packets of 64K bytes of data called datagrams from one host to another. This protocol is unreliable because it does not care if a sent packet is lost, duplicated, or corrupted. The IP relies on higher level protocols to ensure a reliable transmission. The primary tasks of the IP are addressing, routing, and fragmentation. Another duty of the IP is to remove old packets from the network.

151 The IP uses the address carried in the internet header to determine the destination of data packets. Selection of the path for transmission is called routing. Routes are based upon static or dynamic tables. Fragmentation is needed when a block of data is too large to fit the network below IP. If a large block of data arrives at the IP level for transmission over the network, the IP divides the block into smaller fragments before sending them onto the network. When data fragments are received from the network, the IP reassembles the data fragments so as to correspond to the original block.

152 The Transfer Control Protocol ("TCP") is designed to provide a reliable connection between two hosts across different networks using the Internet Protocol. The data sent with TCP/IP is considered to be a stream of bytes, even though the data has to be split into smaller packets due to the underlying protocols. TCP must recover data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. TCP also provides a way for the receiver to govern the amount of data sent by the sender, i.e., it provides flow or congestion control. Another responsibility of the TCP protocol is connection handling, wherein TCP manages a number of connections at the same time.

153 The unit transferred between two host machines using the TCP protocol software is called a segment. Segments are exchanged to establish connections, to transfer data, to send acknowledgments ("ACK") that data was received, to define window sizes, and to close connections. Each segment is divided into two parts, a header and data. The TCP header carries the identification and control information and contains all the necessary information to ensure safe transfer of the data, to handle flow control, and to handle connections. Fig. A1 illustrates a conventional TCP segment with the data and the elements of the header, and is described in detail below.

154 As seen in Fig. A1, the Source Port is a 16-bit number that identifies where the TCP segment originated from within the sending host, and the Destination Port is a 16-bit number that identifies where the TCP segment is destined in the receiving host. The Sequence Number is a 32-bit number identifying the current position of the first data byte in the segment within the entire byte stream for the TCP connection. The

Acknowledgement Number is a 32-bit number identifying the next data byte the sender expects from the receiver.

155 HLEN is the header length, and it is a 4-bit field that specifies the total TCP header length. The field identified as Reserved is a 6-bit field currently unused and reserved for future use. Code bits are single bit flags that notify either the sending host or the receiving host that certain variables are being implemented during the TCP session. The Window is a 16-bit number used by TCP for flow control in the form of a data transmission window size, and it tells the sender how much data the receiver is willing to accept. The Checksum is preferably a 16-bit value that is computed by the TCP sending host, based on the contents of the TCP header and data fields. This 32-bit checksum will be compared by the receiver with the value the receiver generates using the same computation. If the values match, the receiver is assured that the segment arrived intact.

156 Where it is necessary for a TCP sender to notify the receiver of urgent data that should be processed by the receiving application as soon as possible, the Urgent Pointer is used. It is a 16-bit field that tells the receiver when the last byte of urgent data in the segment ends. In order to provide additional functionality, several optional parameters may be used between a TCP sender and receiver. Because options may vary in size, it may be necessary to "pad" the TCP header with zeroes, using the Padding field, so that the segment ends on a 32-bit word boundary as defined by the standard. Although the Data field is not used in some circumstances (e.g., ACK segments with no data in the reverse direction), this variable length field carries the application data from the TCP sender to receiver.

157 Local Area Networks ("LAN"), Metropolitan Area Networks ("MAN"), or Wide Area Networks ("WAN") typically use TCP/IP to transfer data from one computer to another, and Gigabit Ethernet is widely used as the physical medium in these environments. The Gigabit Ethernet, and similar technologies, enable fast transmission of the IP data once put on the wire. However, the prior art implements TCP/IP exclusively in software and is thus unable to utilize the potential increase in transfer rate through implementation of faster TCP/IP protocol processing. Therefore,

what is needed is a method for implementing TCP/IP protocol in both software and hardware.

158 The present invention provides for a method for transmitting a stream of data, comprising a plurality of data segments, from a sending computer to a receiving computer using a Transmission Control Protocol, the method comprising the steps of: (a) initializing a plurality of variables in a data structure, the data structure for use in transmitting the data segments in the data stream from the sending computer to the receiving computer, the plurality of variables in the data structure including: (1) an initial sequence number; and (2) the total number of data segments to be transmitted; and (b) transmitting the data segments using a plurality of microcode driven processing engines. The processing engines preferably comprise a re-transmission engine in the sending computer for determining whether a transmitted segment should be re-sent, an acknowledgement engine in the receiving computer for determining whether a transmitted segment was received by the receiving computer and for transmitting an acknowledgement signal to the sending computer, and an out-of-sequence detector in the receiving computer for detecting whether any the transmitted data segment was received out of sequence, and if so, for re-sequencing the data segments into their correct order. The step of transmitting the data segments comprises: (1) determining a rate at which the data segments are to be sent from the sending computer to the receiving computer; (2) identifying one of the data segments to be sent, assigning a sequence number to the selected data segment, and sending the selected data segment to the receiving computer; (3) initializing a retransmission timer as a function of the time when the selected data segment was transmitted and calculating a time duration value for the retransmission timer; (4) determining whether the time duration of the retransmission timer has expired before receipt of the acknowledgement signal for the selected data segment from the receiving computer and if so, resending the data segment and returning to step (3); and (5) determining whether additional the data segments are to be sent to the receiving computer, and if so repeating steps (1)-(5) until all data segments are successfully transmitted.

159 An object of the present invention is to define a new data structure (TCB) for use in implementing TCP/IP protocol in hardware. A key advantage of the present invention is that it increases the data transfer rate in TCP/IP protocol by using the

TCB data structure to implement TCP/IP protocol through microcode driven processing engines contained on one or more integrated circuit chips, e.g., Application Specific Integrated Circuit (ASIC) chips, instead of implementing TCP/IP protocol entirely through software. Therefore, efficiencies are achieved in the transmission and any necessary re-transmission of data, the acknowledgement that data was received, and in flow and congestion control.

160 The foregoing aspects and attendant advantages of the present invention will become more readily apparent by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

161 Fig. A1 is a block diagram illustrating the elements of a prior art TCP data segment;

162 Fig. A2a illustrates the layout in memory of a TCB data structure according to a preferred embodiment of the present invention;

163 Fig. A2b is a flow chart illustrating Reorder Buffer (ROB) update and processing;

164 Figs. A3a-A3g list the preferred transmit variables, receive variables, and re-order variables used in a plurality of algorithms needed to implement the TCP protocol;

165 Fig. A4 is a flow diagram illustrating the steps for transmitting a stream of data, comprising a plurality of data segments, from a sending computer to a receiving computer according to a preferred embodiment of the present invention;

166 Fig. A5 is a block diagram illustrating an ASIC having microcode driven processing engines for TCP/IP processing according to the present invention;

167 Fig. A6 illustrates how Fibre Channel frame to Gigabit Ethernet frame processing is performed in the ASIC of Fig. A5; and

168 Fig. A7 illustrates how Gigabit Ethernet frame to Fibre Channel frame processing is performed in the ASIC of Fig. A5.

169 The present invention increases the data transfer rate in TCP/IP protocol by defining a new data structure (TCB) that implements TCP/IP through software microcode driven processing engines. TCB preferably occupies a contiguous block of memory of predetermined size. Fig. A2 illustrates an exemplary layout in memory of the TCB data structure according to a preferred embodiment of the present invention. Each TCB has associated with it pointers to three contiguous blocks of memory that store a "transmit, pending & acknowledgement" pending queue (TXP), a "received data segments processing" pending queue (RXP), and a buffer for reordering data segments if they are received out of sequence.

170 In addition, there are two queues called the transmit queue (TX) and the receive queue (RX) that are shared by all the TCP sessions. The transmit queue stores pointers to data that is to be transmitted to the receiving host and the RX queue stores pointers to data that has been received by the receiving host and is awaiting processing in memory. The Transmit Queue contains: a pointer that is maintained to point to the Transmit Pending Queue; a Write Index – TxQWrIndex; a Read Index – TxQRdIndex; and Retransmission Index – TxQRtxIndex. Finally, the Receive Queue contains a pointer that is maintained to points to the Start of the Receive Queue.

171 The Transmit Pending Queue is a contiguous block of memory of whose size determines the maximum number of pending entries that can be handled. This queue is maintained as a circular queue. Each entry consists of a pointer to the actual data segment, total length of the data segment, and Starting Sequence Number. Since memory is best utilized if the data is read in 64 byte chunks, each 64 byte chunk contains as many as four entries. To access this queue, three pointers identified as RdIndex, RtxIndex and WrtIndex, are maintained. Initially, these pointers are pointing to zero.

172 The functionality of the TXP is described by the following steps:

173 1. When a data block is received from the FC layer and is meant to be transmitted, it is put in the Transmit Pending Queue position given by the WrIndex. Update the WrIndex afterward.

174 2. When the window size of the TCP session is sufficient to transmit 1 or more data segments, the data segments are picked up starting from the RdIndex position and put into the TX queue. Update the RdIndex after that.

175 3. When an acknowledgement is received, then increment the RtxIndex pointer beyond those data segments which have been covered by the acknowledgement.

176 4. Whenever there is a timeout experienced because of a missing acknowledgement, put the data segment indicated by the RtxIndex pointer into the TX queue and reinitialize the timer with the next RTT value. (Exponential backoff scheme may be used each time a retransmission is performed for the same data segment.)

177 The functionality of the RXP is described by the following steps:

178 1. Any data segment on the RX queue that is identified to be meant for a particular TCB should be put into the receive pending queue (RXP) of that particular TCB. The receive sequence variables should be updated if the data received was in-order.

179 2. The items in the receive pending queue store data for this particular TCP session, but they may not be in order. Also some of the data segment may contain only part of the FC data block since TCP may have broken up the FC data block.

180 3. Reordering of items in the receive pending queue has to be done. To reorder packets, the packets received are stored in a linked list form. Sequence Number is used as the Key. When packets come out of order, they are stored in ascending order of Sequence Numbers. Fig. A2b is a flow chart that illustrates Reorder Buffer (ROB) update and processing according to the present invention.

181 4. Whenever more data is obtained in proper sequence, an ACK must be generated to acknowledge the additional data received.

182 5. The new window has to be calculated depending on how much of the received data is sitting in memory and how much has already been transferred to the FC end. Thus, whenever an FC data block is transmitted to the receiver, this information should be used to update the window for TCP to release on the next transmission of an ACK. These contiguous blocks of memory could be allocated as and when the TCB gets used to set up an actual TCP session.

183 Figs. A3a-A3g list the preferred transmit variables, receive variables, and re-order variables used in a plurality of algorithms needed to implement the TCP protocol, and Fig. A4 is a flow diagram illustrating a process 40 for transmitting a stream of data, comprising a plurality of data segments, from a sending computer to a receiving computer according to a preferred embodiment of the present invention.

184 As illustrated in Figs A3a-A3g, there are three timers, the Retransmission Timer, the Idle Timer and the Persist Timer. Whenever the Retransmission Timer runs out, it should put an entry of a pointer to the TCB into the tasks queue for the retransmission process to work on. The Idle Timer is used to check for idle session and, if the timer runs out, then a dummy packet is sent to keep the session alive.

185 The Round Trip TCB Variables needed to calculate the Round Trip Delay include: a Current Retransmission Value-16 bit value (rxtcur); Log(2) of exponential backoff-16 bit value (rxtshift); a Round Trip time-16 bit value (rtt); a Minimum RTT value-16 bit value, which is calculated from the route characteristics (rttmin); a Current RT Sequence Number – 32 bit value (rtseq); a Smoothed Round Trip time-16 bit value – (srtt); and a Variance in Round Trip – 16 bit value (rttvar).

186 The TCP and IP Header parameters include: an IPDA (32 bit value) IP Destination Address; an IP Diffserv (8 bits) that specifies the Type of Service; and source and destination port values (16 bit values) to store the TCP source and destination port numbers.

187 Send Sequence Variables include: Send Unacknowledged (32 bit value) indicating the oldest unacknowledged sequence number; Send Next (32 bit value) indicating the sequence number that will be used for the next transmission; Send Urgent Pointer (32 bit value) indicating the 16-bit offset in the data where the urgent

data is sitting, which may be needed to communicate the other end that there is some urgent data in the packet transmitted; Send Window (16-bit value); Send Maximum-Highest Sequence Number Sent (32-bit value), which will be used for retransmission; and an Initial Sequence Number (32 bit value). The Receive Sequence Variables include: a Receive Window (16 bit value); a Receive Next (32 bit value); a Receive Urgent Pointer (32 bit value); and an Initial Receive Sequence Number (32 bit value).

188 When the receiving computer wants to send a stream of data to the receiving computer according to process 40 illustrated in Fig. A4., it establishes a TCP session. Then at step 41, the TCB Variables are initialized by software, preferably in the sending computer, in accordance with what is indicated in Figs. A3a-A3g. For instance, the Valid Flag variable indicates whether this is a valid TCB, and when a TCB is assigned this flag will be set to 1. However, care must be taken to clear certain fields, which might be there from a previous connection. Moreover, a Connection State variable reflects the current state of the TCP layer, and can be one of the following states: STATE_CLOSED; STATE_LISTEN; STATE_SYN_SENT; STATE_SYN_RECEIVED; STATE_ESTABLISHED; STATE_CLOSE_WAIT; STATE_FIN_WAIT_1; STATE_CLOSING; STATE_LAST_ACK; STATE_FIN_WAIT_2; STATE_TIME_WAIT. The Connection State is modified by the TCP layer and will be read by all the other blocks to figure out the state of a particular TCP session. Other variables initialized are those regarding an initial sequence number for the segments being transmitted and the total number of segments being transmitted.

189 Referring again to Fig. A4, one the TCB data structure is initialized, at step 42, the data segments in the data stream are transmitted using a plurality of microcode driven processing engines, according to steps 43-48. The microcode driven engines comprise a re-transmission engine in said sending computer for determining whether a transmitted segment should be re-sent, an acknowledgement engine in said receiving computer for determining whether a transmitted segment was received by said receiving computer and for transmitting an acknowledgement signal to said sending computer, and an out-of-sequence detector in said receiving computer for detecting whether any said transmitted data segment was received out of sequence, and if so, for re-sequencing said data segments into their correct order.

190 A rate at which the data segments are to be sent is determined, at step 43. The congestion control engine in used in determining this rate and is one of the microcode driven engines, and the receive sequence variable described above are calculated and used. The entire congestion control scheme is implemented in Microcode. User can configure this according to the environment. Current implementation is as follows: when RTO expires the congestion window is converted into 2*MSS (Maximum Segment Size). The recovery is done using a slow start process where in the congestion window is doubled every time an ACK is received. Duplicate ACK packets will be used to change the Congestion window size to 2*MSS. The first segment is re-transmitted. All these processes are done in hardware in the microcode driven engine.

191 At step 44, a data segment is identified to be sent, a sequence number is assigned to the data segment and the data segment is sent. In this step the send sequence variables discussed above are calculated and used, and the transmit pending queue is utilized. For instance, whenever a new data block arrives from a lower protocol layer, one or more nodes are added to the transmit pending queue to allow data transmission. Adding a Node consists of these steps:

192 1. If (WrtIndex+1)%MAX_TX_PENDING_ENTRIES == RtxIndex, then this is over flow condition;

193 2. Memory Offset to Write = Queue Base Pointer + ((WrtIndex*16)/64)*64;

194 3. Do a 64-byte block read using the above offset;

195 4. Add Offset = ((WrtIndex*16)%64);

196 5. Now pointer to the data block and the total length can be added using the Add Offset;

197 6. Write back the 64-byte block using the "Memory Offset to Write";

198 7. Increment WrtIndex = (WrtIndex + 1)%MAX_TX_PENDING_ENTRIES.

199 Deleting a node consists of these steps:

200 1. If RdIndex == WrtIndex, then the block is empty, nothing to read;

201 2. Memory Offset to Read = Queue Base Pointer + ((RdIndex*16)/64)*64;

202 3. Do a 64-byte block read using the above offset;

203 4. Read Offset = ((RdIndex*16)%64);

204 5. Read the data block pointer and the total length;

205 6. If the action is delete this node, then RdIndex has to be changed;

206 7. RdIndex = (RdIndex + 1)%MAX_TX_PENDING_ENTRIES.

207 Once a segment has been sent, the retransmission engine is used to calculate the retransmission variables and set a retransmission timer at step 45. The retransmission algorithm for the retransmission timer is as follows. Every time a segment is sent, the TCP Engine starts a timer and waits for an acknowledgement. If the timer expires before acknowledgement is received, TCP Engine assumes the segment was lost or corrupted and retransmits it. The re-transmission timer value is calculated as

208 SRTT = 0.9 * OLD_SRTT + 0.1 * MRTT

209 RTO = 2 * SRTT

210 Where RTO → Retransmission Time out, SRTT → Smoothed Round Trip Time,

211 MRTT → Measured Round Trip Time.

212 All of these calculations are done in Protocol Processing Engines in Hardware. At step 46, it is determined whether the retransmission time has expired before an ACK is received on the selected segment, and if so, the segment is resent at step 47 and steps 45 and 46 are repeated. If not, then the ACK was sent. Moreover, it is

determined whether all data segments in the data stream were sent at step 43, and if not steps 43-48 are repeated until all data segments are successfully sent.

213 Date segment acknowledgement functions are performed in a microcode driven processing engine. Whenever an ACK is received for the TCB, the send window parameters change, and so additional window space may become available. When an ACK is received then Window is forwarded appropriately. Every time an ACK packet is received a sliding window (described in detail below) is moved and next set of packets is sent. Upon receiving segments properly (if the check sum is OK), an ACK packet is transmitted with an updated sequence number. If the receive buffer is full an ACK packet with window size 0 is sent.

214 TXP and ACK respectively keep track of where the next data segment to be transmitted is to be stored in memory, which is the next data segment to be put in the transmit queue (TX) and which is the next data segment to be retried in case the TCP session experiences a timeout while waiting for an ACK. The following actions are taken on an ACK reception.

215 1. When an ACK is received some data segments will have to removed from the Transmit Pending Queue by moving the RtxIndex beyond all those data segments that have been properly acknowledged.

216 2. The send sequence variables have to be properly updated using the highest sequence number acknowledged and using the newly indicated window size.

217 3. If there is data along with the ACK, the receive sequence variables should also be updated (if the data is in order), and the data segment should be stored in memory and a pointer to the same should be added into the RXP queue of the TCB.

218 Through the transmission of data segments, one or more segments may be received out of sequence. The out of sequence detector is the microcode driven processing engine used in such a situation. For instance, lets us assume a seven segments with their byte streams as follows

219

	Seg 2, 1400	Seg 3, 2000	Seg 4, 2400	Seg 5, 2800	Seg 6, 3200	Seg 7, 4000

220 Segment 1 arrives at the receiver which acknowledges it by sending an ACK segment with ACK=1000 and WIN=1600 (Window size 1600). Since ACK+WIN=2600 the sender can send segments 2,3 and 4. Segments 2 and 3 are sent. Segment 3 has arrived but segment 2 has been delayed. The receiver sends another ACK segment with ACK=1000 and WIN=1600. The sender sends segment 4. Segment 4 arrives but segment 2 is still outstanding. Again the receiver sends ACK=1000, WIN=1600. However the sender cannot send segment 5 since it would take the sent sequence number to 2800 which is greater than ACK+WIN. The sender cannot do anything further unless Segment 5 is split into smaller segments to use up the remaining part of the window. Segment 2 reaches the receiver and the receiver announces a new ACK value. The delayed segment 2 now reaches the receiver which sends ACK=2400 and WIN=1600 allowing the sender to send up to 4000.

221 The initial values of the sequence numbers are exchanged during the connection establishment sequence. This flow control mechanism shown above is usually called a sliding window protocol. When packets are received out of Sequence it is stored in the Re-order buffer. These packets are sorted and stored in the external memory.

222 The Hardware implementation of TCP Engine is done using a set of protocol processing engines, which are microcode driven. Initial parameters for processing are setup by the software through an external CPU. Calculations for the algorithms are done using the above described microcode driven engines. This method provides flexibility for changes.

223 Fig. A5 is a block diagram illustrating an ASIC having microcode driven processing engines for TCP/IP processing according to the present invention;

224 Fig. A6 illustrates how Fibre Channel frame to Gigabit Ethernet frame processing is performed in the ASIC of Fig. A5; and

225 Fig. A7 illustrates how Gigabit Ethernet frame to Fibre Channel frame processing is performed in the ASIC of Fig. A5.

226 Thus, the invention relates, in one embodiment to a method for transmitting a stream of data, comprising a plurality of data segments, from a sending computer to a receiving computer using a Transmission Control Protocol, the method comprising the steps of initializing a plurality of variables in a data structure, the data structure for use in transmitting the data segments in the data stream from the sending computer to the receiving computer, the plurality of variables in the data structure including an initial sequence number; and the total number of data segments to be transmitted. The method further includes transmitting the data segments using a plurality of microcode driven processing engines, the processing engines comprising, a re-transmission engine in the sending computer for determining whether a transmitted segment should be re-sent, an acknowledgement engine in the receiving computer for determining whether a transmitted segment was received by the receiving computer and for transmitting an acknowledgement signal to the sending computer, a congestion control engine for determining a rate at which data segments are to be sent from the sending computer to the receiving computer, and an out-of-sequence detector in the receiving computer for detecting whether any the transmitted data segment was received out of sequence, and if so, for re-sequencing the data segments into their correct order. The step of transmitting the data segments comprises (1) determining the rate at which the data segments are to be sent from the sending computer to the receiving computer, (2) identifying one of the data segments to be sent, assigning a sequence number to the selected data segment, and sending the selected data segment to the receiving computer, (3) initializing a retransmission timer as a function of the time when the selected data segment was transmitted and calculating a time duration value for the retransmission timer, (4) determining whether the time duration of the retransmission timer has expired before receipt of the acknowledgement signal for the selected data segment from the receiving computer and if so, resending the data segment and returning to step (3) and (5) determining whether additional the data segments are to be sent to the receiving computer, and if so repeating steps (1)-(5) until all data segments are successfully transmitted.

227 In one embodiment, the microcode driven re-transmission engine comprises a re-transmission timer algorithm for calculating a re-transmission timer value, the

algorithm depending upon a smoothed round trip time value and a measured round trip time value.

228 In another embodiment, the microcode driven acknowledgement engine comprises at least one algorithm, the at least one algorithm configured to send the acknowledgement signal when the data segment is properly received by the receiving computer; and move a sliding window and instruct sending the next data segment in the stream of data when the sending computer receives the acknowledgement signal; and send the acknowledgement signal, wherein a window size header element is zero when a receive buffer is full.

229 In another embodiment, the microcode driven congestion control engine comprises a plurality of algorithms, the plurality of algorithms including a slow start algorithm for doubling the size of a congestion window when each the acknowledgement signal is received by the sending computer, an algorithm for changing the size of the congestion window when a re-transmission timer expires, the algorithm depending upon a maximum segment size, and an algorithm for changing the size of the congestion window when a duplicate acknowledgement signal is received by the sending computer, the algorithm depending upon the maximum segment size.

230 In another embodiment, the microcode driven out-of-sequence detector comprises a flow control algorithm for implementing a sliding window protocol.

231 In one embodiment, there is disclosed a method for transmitting a stream of data, having a plurality of data segments, from a sending computer to a receiving computer using a Transmission Control Protocol. The method includes the steps of initializing a plurality of variables in a data structure that will be used in transmitting the data segments and transmitting the data segments using a plurality of microcode driver processing engine that each perform an algorithm to process the initialized variables and to update those variables.

232 The implementation example of a method for implementing TCP/IP protocol in hardware described in the text above was chosen as being illustrative of the best mode of the present invention. The preferred embodiment of the present invention

described above is illustrative of the principles of the invention and is not intended to limit the invention to the particular embodiment described. Accordingly, while the preferred embodiment of the present invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

233 Thus, while this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

CLAIMS

What is claimed is:

1. In a target device, a method for partially reordering a plurality of data packets transmitted from a source device, said source device being coupled to said target device via a computer network, comprising:

receiving a first set of data packets from said transmitted device;

ascertaining whether said first set of data packets represents a set of data packets that said target device expects to receive next; and

if said first set of data packets does not represents said set of data packets that said target expects to receive next, storing said first set of data packets in a memory buffer of said target device, said storing including arranging said first set of data packets in said memory buffer such that data packets in said memory buffer, including said first set of data packets, are in order in said memory buffer .

2. The method of claim 1, wherein each packet of said plurality of data packets is associated with a sequence number, said sequence number determining a spatial relationship of said each packet relative to other packets in said plurality of data packets.

3. The method of claim 2, wherein said memory buffer is comprised of a plurality of storage structures for storing data packets of said plurality of data packets by their sequence numbers.

4. The method of claim 3, wherein said each storage structure of said plurality of storage structures is configured to store a pointer to a memory location different from said memory buffer, said memory location being employed to store at least part of a packet received by said target device.

5. The method of claim 4, including allocating in advance at least one storage structure of said plurality of storage structures for storing a given data packet, said

given data packet represents a data packet that said target device expects to receive next from said source device.

6. The method of claim 5 wherein a data packet received at said target device after said allocating is discarded if said data packet received at said target device after said allocating does not represent said data packet that said target device expects to receive next from said source device.

7. The method of claim 4, including allocating in advance a set of storage structures of said plurality of storage structures for storing a given plurality of data packets, said given plurality of data packets represents data packets that said target device expects to receive next from said source device.

8. The method of claim 7 wherein data packets received at said target device after said allocating are discarded if said data packets received at said target device after said allocating do not represent said data packets that said target device expects to receive next from said source device.

9. In a target device, an arrangement for partially reordering a plurality of data packets transmitted from a source device, said source device being coupled to said target device via a computer network, comprising:

means for receiving a first set of data packets from said transmitted device;

means for ascertaining whether said first set of data packets represents a set of data packets that said target device expects to receive next; and

if said first set of data packets does not represents said set of data packets that said target expects to receive next, means for storing said first set of data packets in a memory buffer of said target device, said storing including arranging said first set of data packets in said memory buffer such that data packets in said memory buffer, including said first set of data packets, are in order in said memory buffer .

10. In a networked device, an arrangement for partially reordering packets received from another networked devices, each of said packets being associated with a sequence number, said arrangement comprising a receive transmission control block

associated with a data flow, said receive transmission control block including a plurality of buffers, each of said plurality of buffers being configured for storing at least one pointer to a memory location, said memory location being different from said plurality of buffers, said memory location being configured to store at least a portion of a packet received from said another networked device, wherein said packets received from said other networked devices are stored in first memory locations associated with said plurality of buffers if said packets received from said other networked devices are received out of order according to their sequence numbers, said packets received from said other networked devices are stored in order among themselves in said first memory locations until a plurality of packets stored in said first memory locations is in order, said plurality of packets, once in order, are transmitted from said first memory locations to said networked device.

11. The networked device of claim 10, including logic for allocating in advance at least one memory location in said first memory for storing at least one packet, said at least one packet represents a packet that said networked device expects to receive next from said another networked device.

12. The networked device of claim 10, including logic for allocating in advance a plurality of memory locations in said first memory for storing a given plurality of packets, said given plurality of packets represent packets that said networked device expects to receive next from said another networked device.

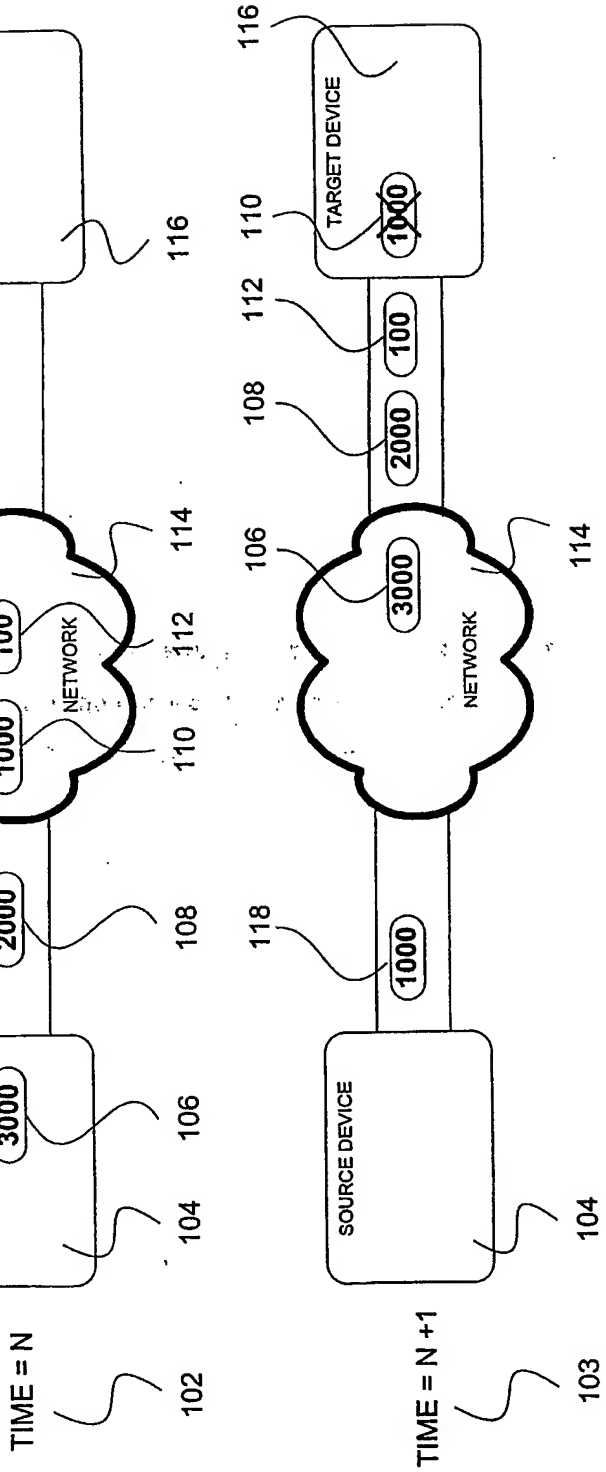


FIG. 1A (PRIOR ART)

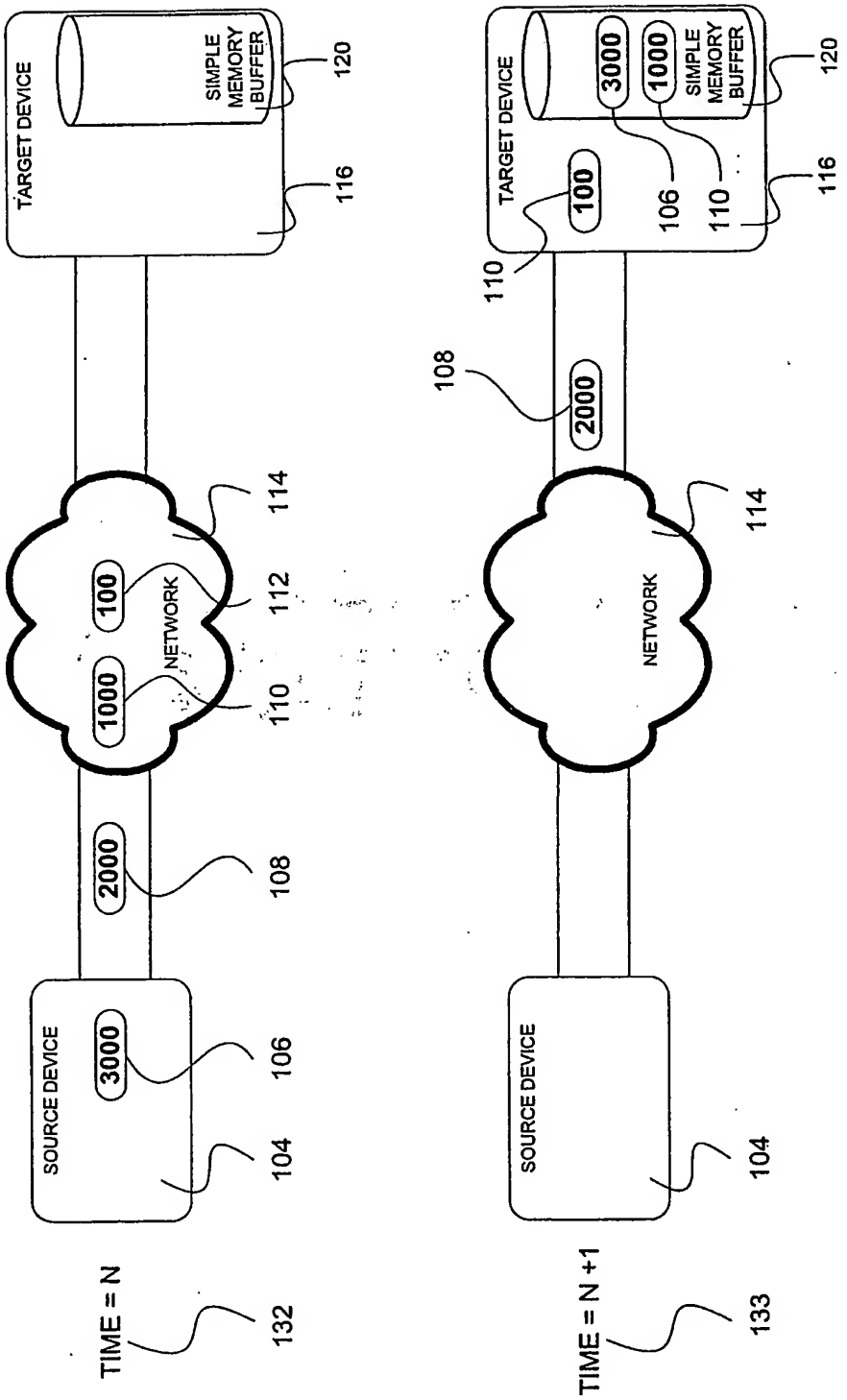


FIG. 1B (PRIOR ART)

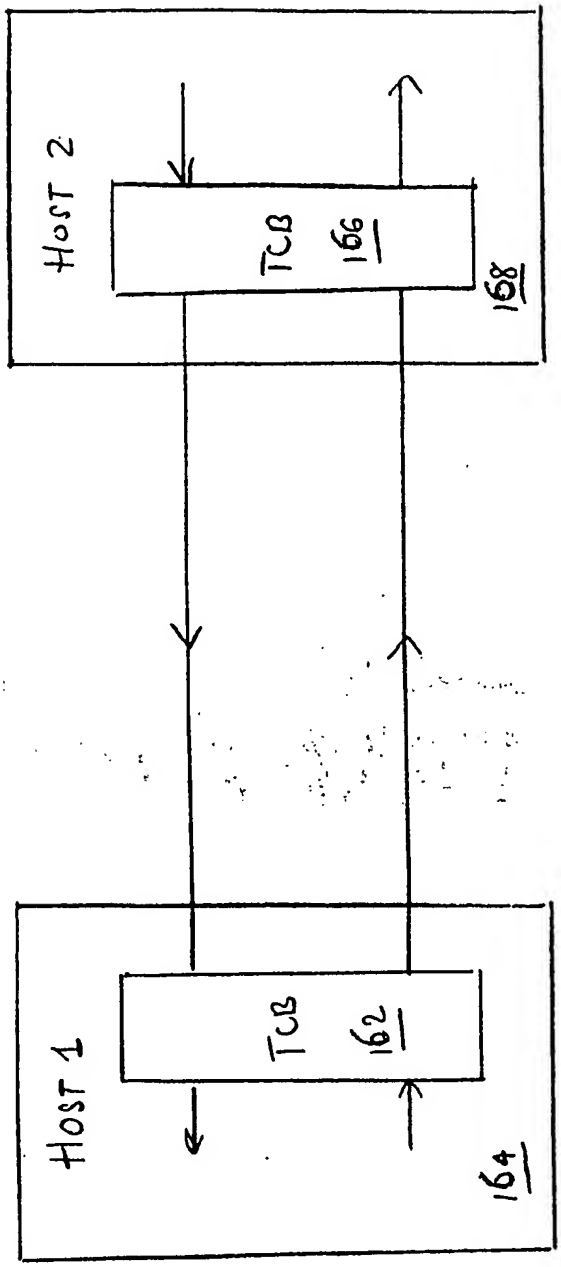


FIG. 1C

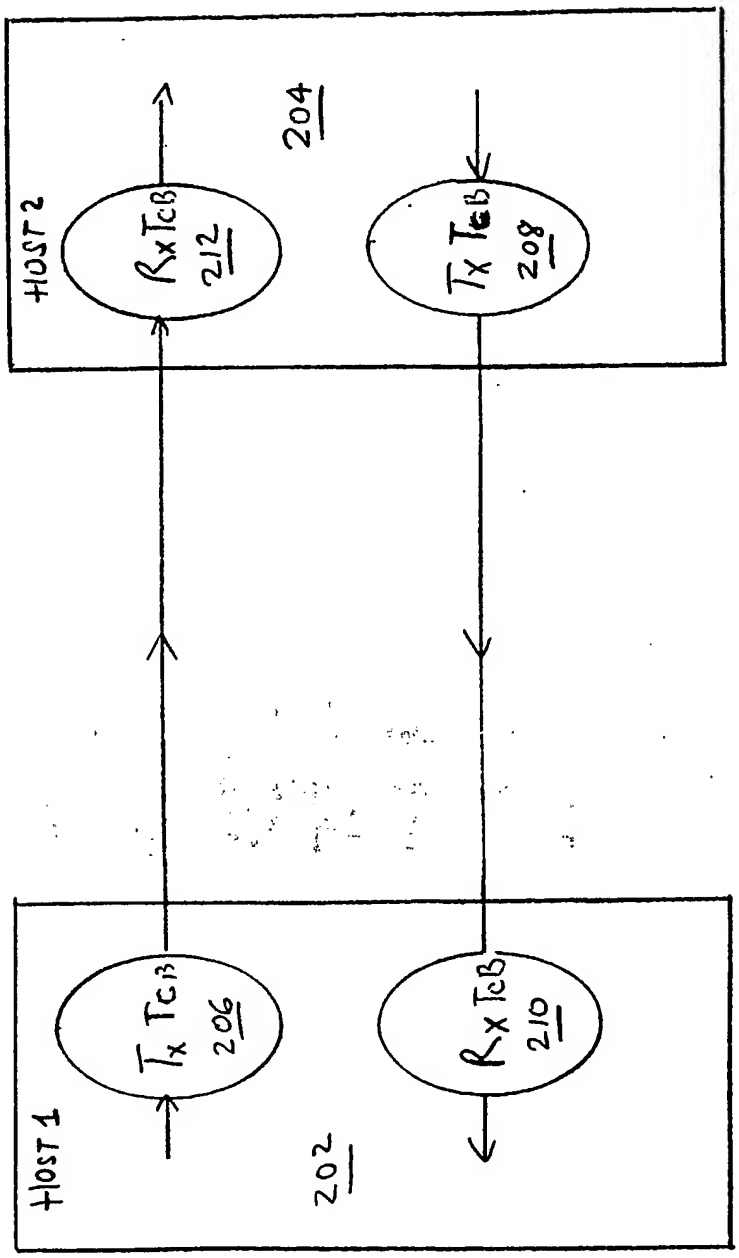


Fig 2

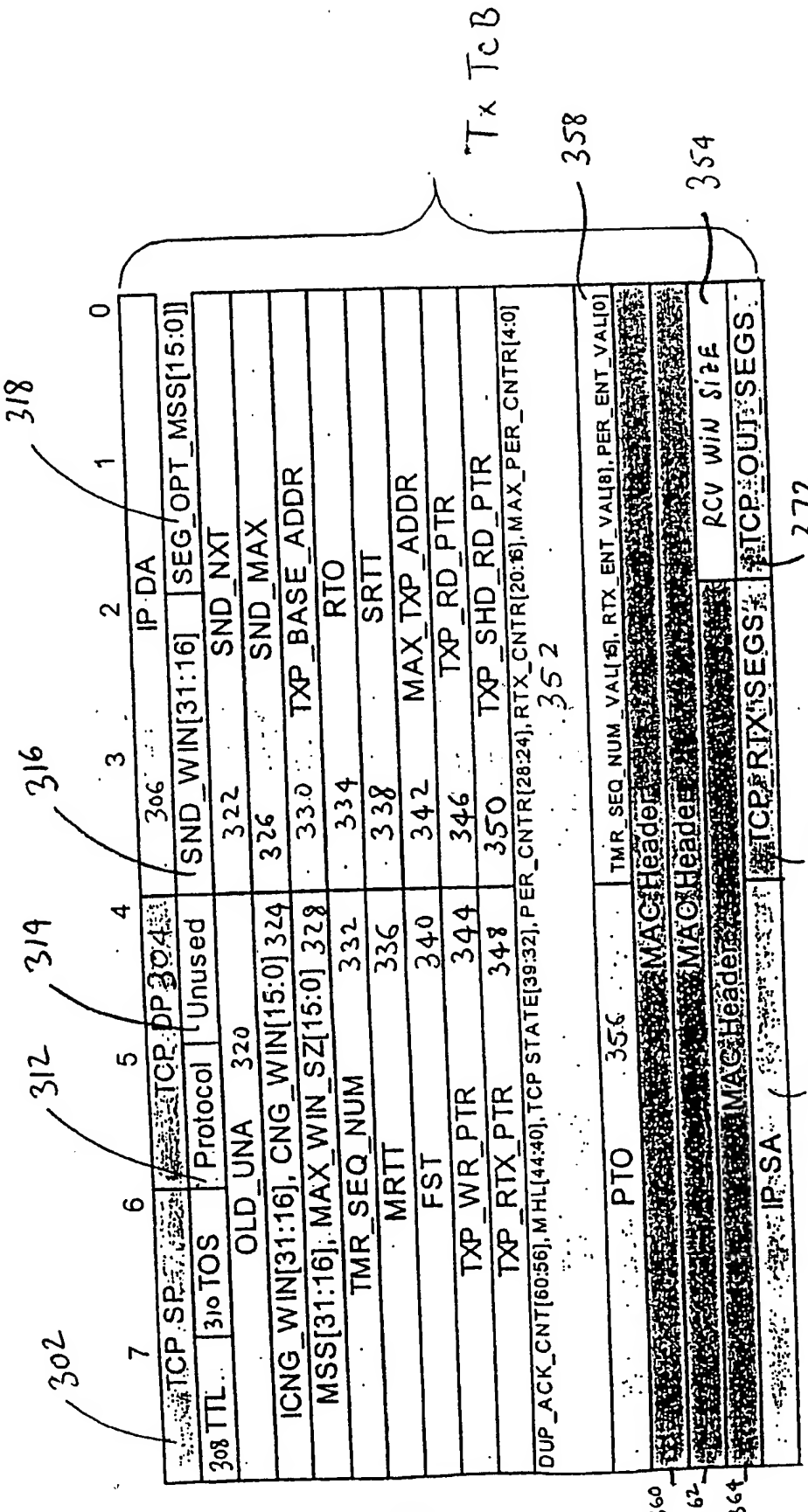

Tx TCB
358
354
372
370
366
302
312
319
316
318
TCP SP
TCP DP 304
IP DA
306
TTL 308
TOS 310
Protocol
Unused
SND_WIN[31:16]
SEG_OPT_MSS[15:0]
OLD_UNA 320
SND_NXT 322
ICNG_WIN[31:16], CNG_WIN[15:0] 324
SND_MAX 326
MSS[31:16], MAX_WIN_SZ[15:0] 328
TXP_BASE_ADDR 330
TMR_SEQ_NUM 332
RTO 334
MRTT 336
SRTT 338
FST 340
MAX_TXP_ADDR 342
TXP_WR_PTR 344
TXP_RD_PTR 346
TXP_RTX_PTR 348
TXP_SHD_RD_PTR 350
DUP_ACK_CNT[60:56], MHL[44:40], TCP STATE[39:32], PER_CNTR[28:24], RTX_CNTR[20:16], MAX_PER_CNTR[4:0]
352
PTO 356
TMR_SEQ_NUM_VAL[15], RTX_ENT_VAL[8], PER_ENT_VAL[0]
MAC Header
MAC Header
MAC Header
RCV WIN SIZE
IP SA
TCP_RTX_SEGS
TCP_OUT_SEGS
360
362
364
7
6
5
4
3
2
1
0

Fig. 3

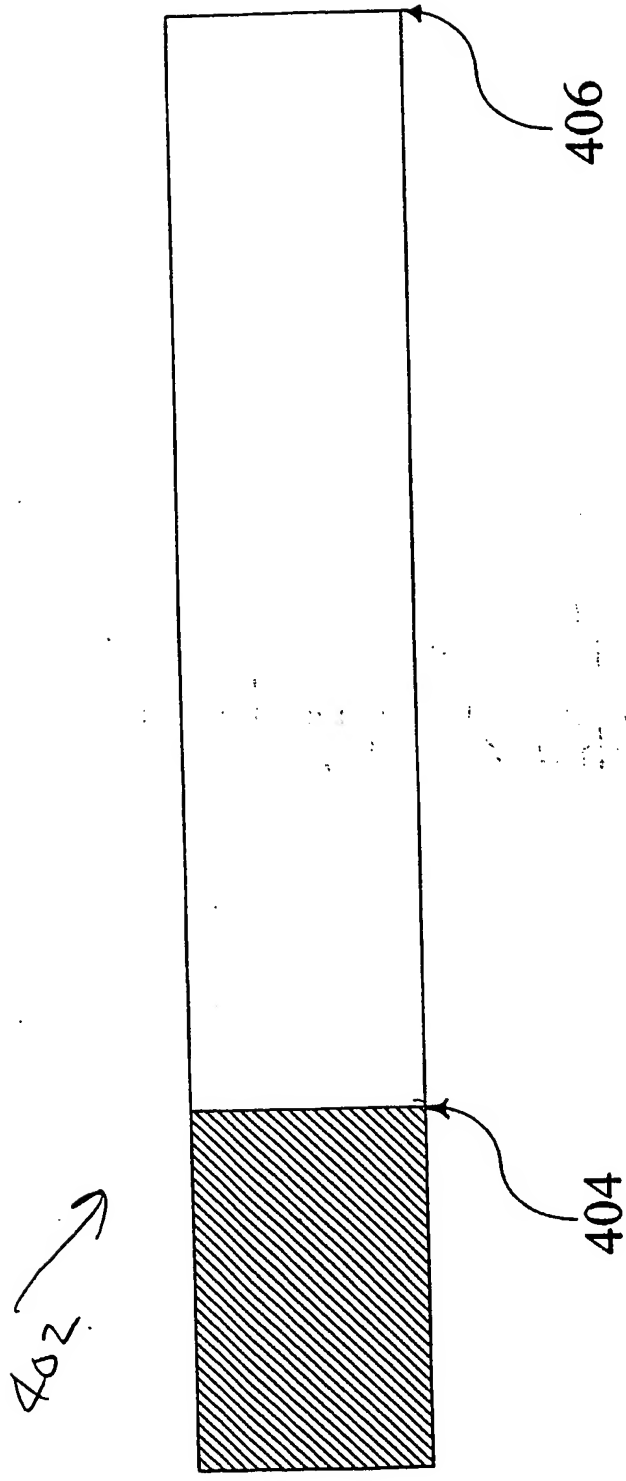


Fig. 4

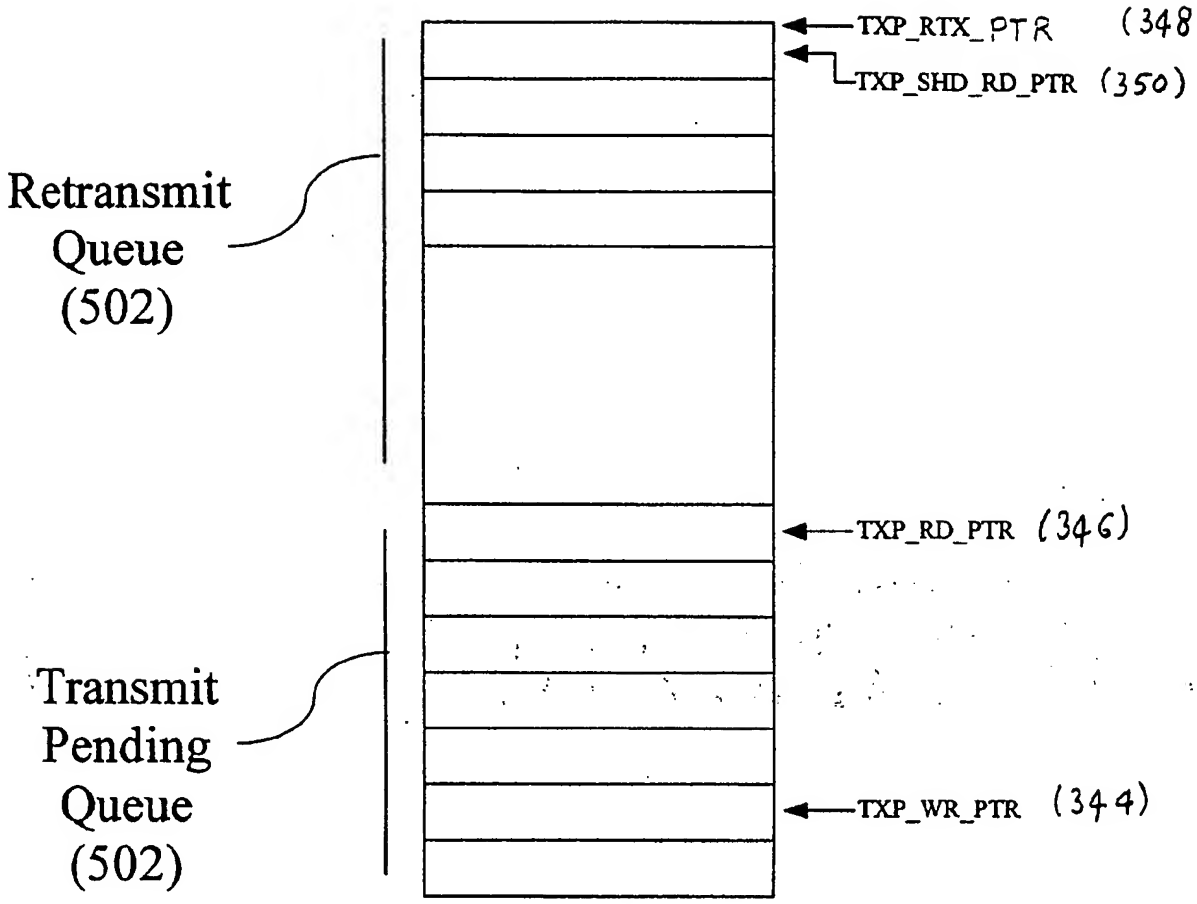

TXP_RTX_PTR (348
TXP_SHD_RD_PTR (350)
Retransmit Queue (502)
TXP_RD_PTR (346)
Transmit Pending Queue (502)
TXP_WR_PTR (344)

Fig. 5

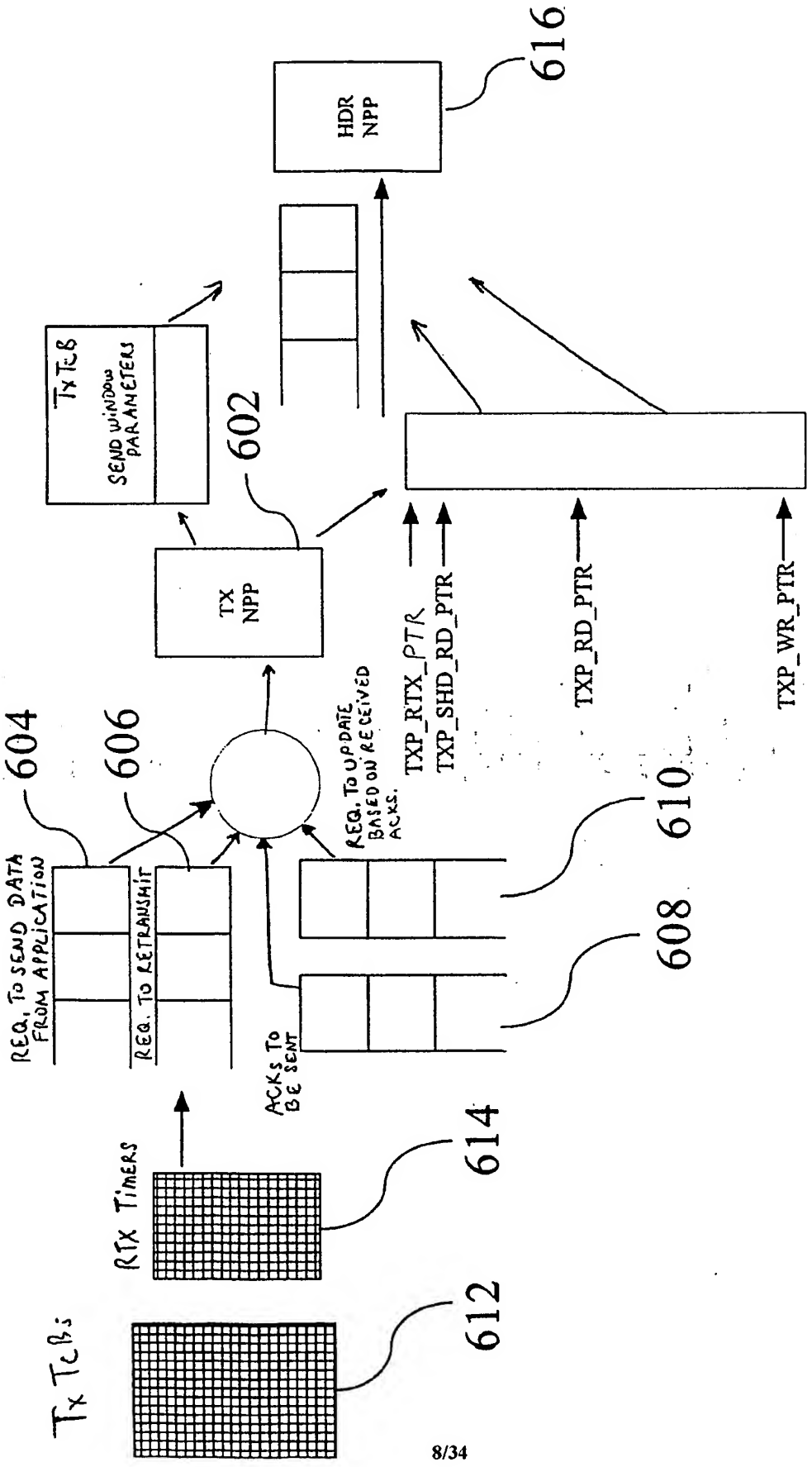

HDR NPP
616
TX TCB
SEND WINDOW PARAMETERS
602
TX NPP
604
606
REQ. TO SEND DATA FROM APPLICATION
REQ. TO RETRANSMIT
REQ. TO UPDATE BASED ON RECEIVED ACKS.
ACKS TO BE SENT
TXP_RTX_PTR
TXP_SHD_RD_PTR
TXP_RD_PTR
TXP_WR_PTR
610
608
RTX TIMERS
614
TX TCBs
612

Fig. 6

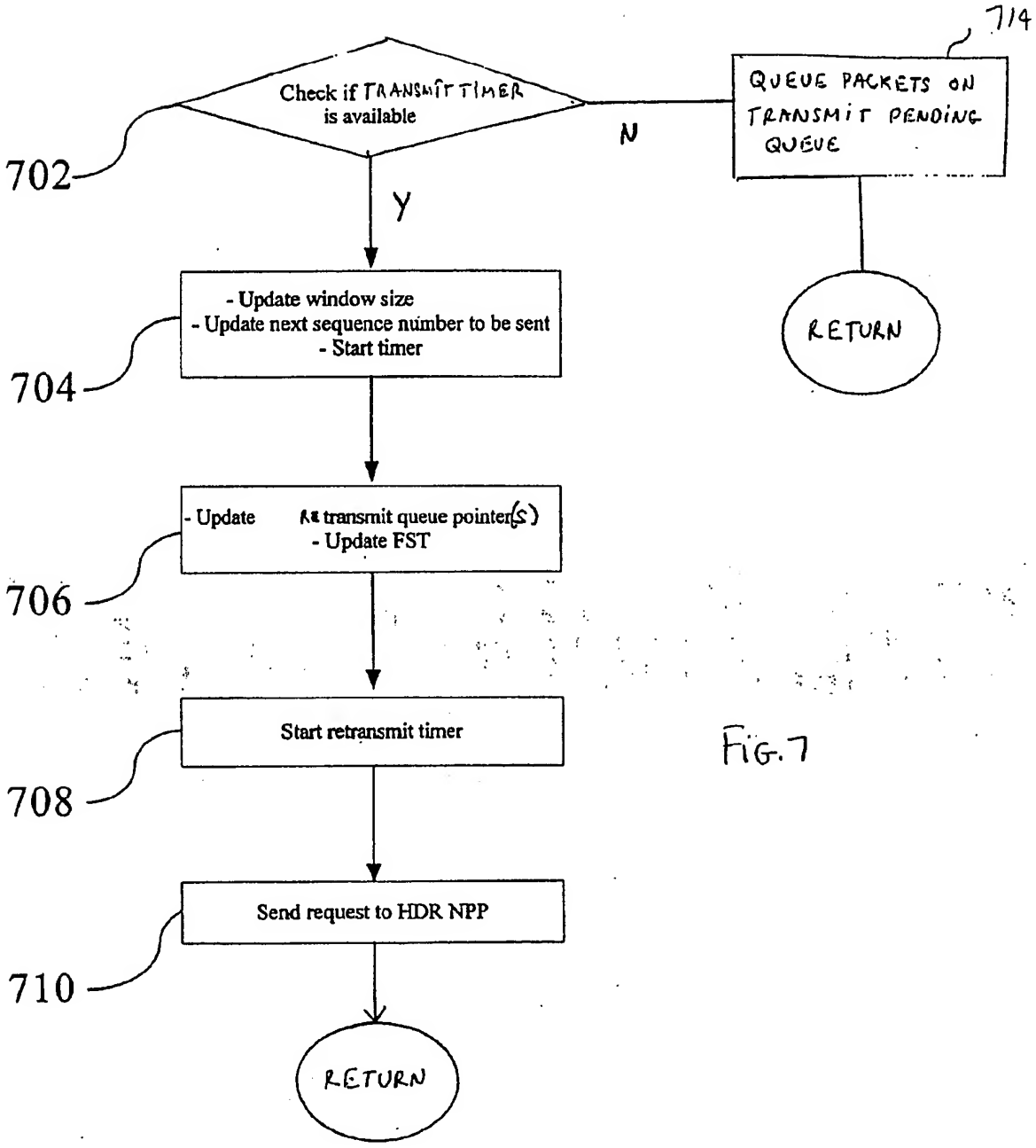

Check if TRANSMIT TIMER is available
702
N
Y
714
QUEUE PACKETS ON TRANSMIT PENDING QUEUE
RETURN
- Update window size
- Update next sequence number to be sent
- Start timer
704
- Update transmit queue pointer(s)
- Update FST
706
Start retransmit timer
708
Send request to HDR NPP
710
RETURN

FIG. 7

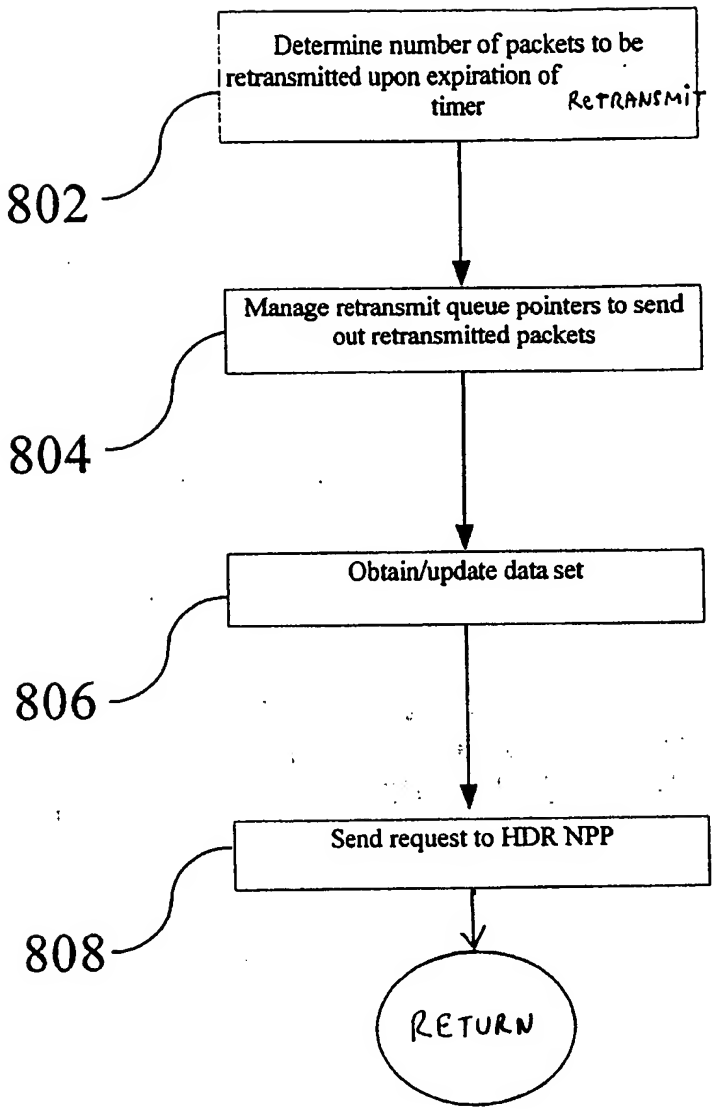

Determine number of packets to be retransmitted upon expiration of timer
RETRANSMIT
802
Manage retransmit queue pointers to send out retransmitted packets
804
Obtain/update data set
806
Send request to HDR NPP
808
RETURN

Fig. 8

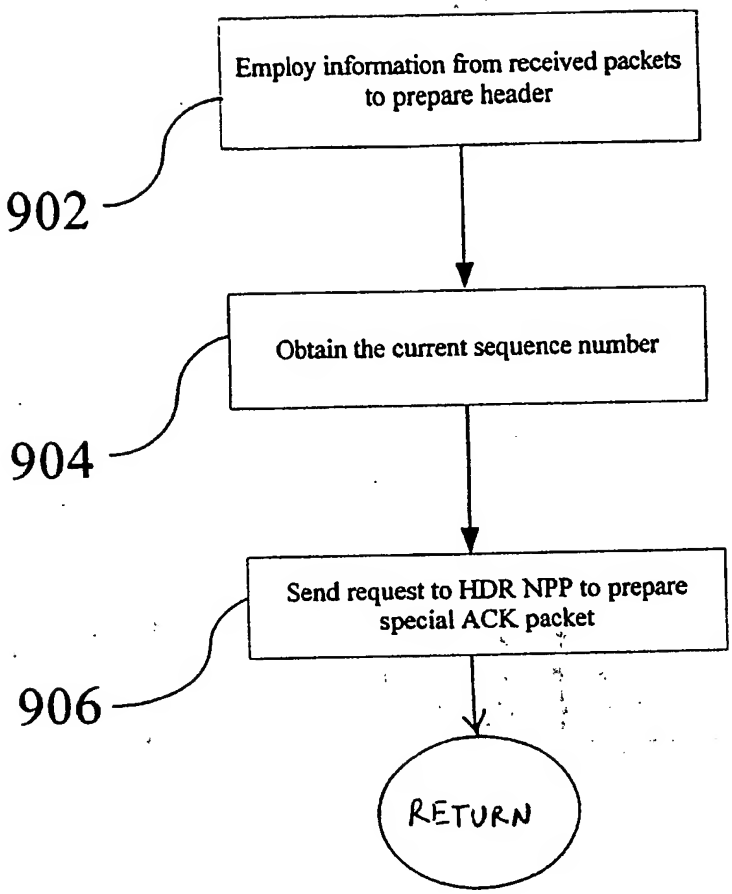

Employ information from received packets to prepare header
902
Obtain the current sequence number
904
Send request to HDR NPP to prepare special ACK packet
906
RETURN

Fig. 9

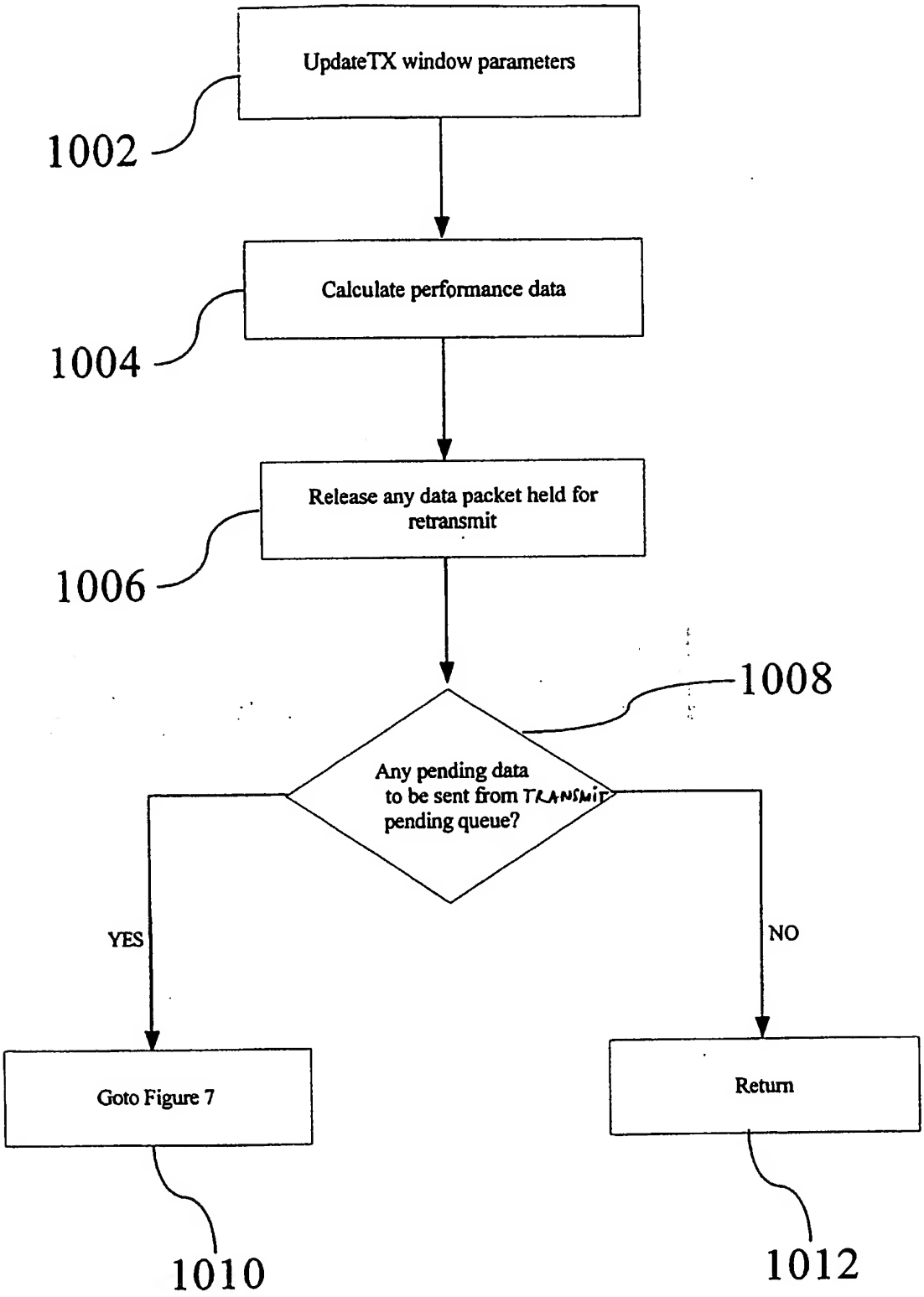

UpdateTX window parameters
1002
Calculate performance data
1004
Release any data packet held for retransmit
1006
1008
Any pending data to be sent from TRANSMIT pending queue?
YES
NO
Goto Figure 7
Return
1010
1012

Fig. 10

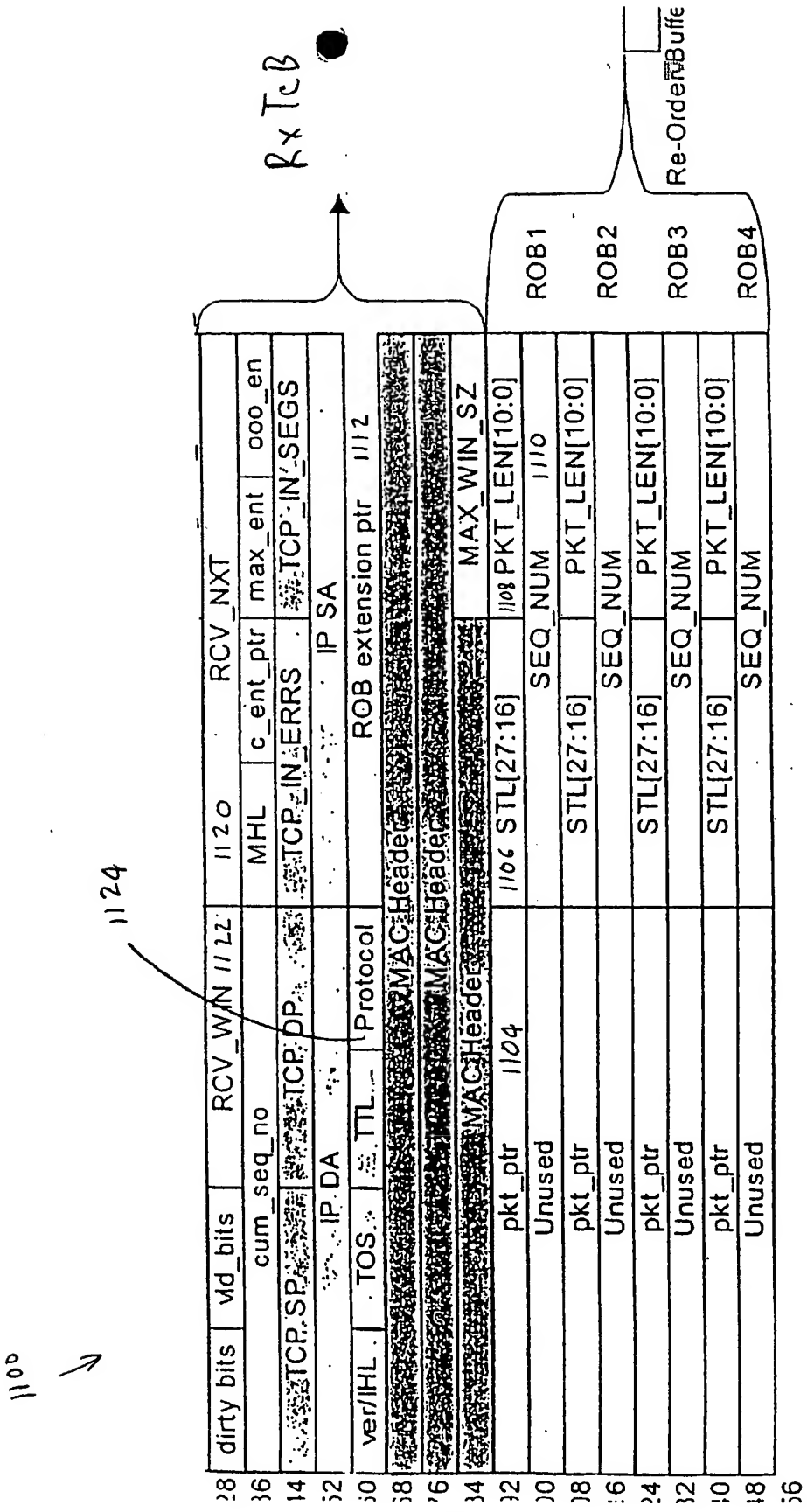

1100
1124
Rx TCB
dirty bits
vld_bits
RCV_WIN 1122
1120
RCV_NXT
cum_seq_no
MHL
c_ent_ptr
max_ent
ooo_en
TCP_SP
TCP_DP
TCP_IN_ERRS
TCP_IN_SEGS
IP_DA
IP SA
ver/IHL
TOS
TTL
Protocol
ROB extension ptr 1112
MAC Header
MAC Header
MAC Header
MAX_WIN_SZ
pkt_ptr 1104
1106 STL[27:16]
1108 PKT_LEN[10:0]
SEQ_NUM 1110
Unused
ROB1
ROB2
ROB3
ROB4
Re-Order Buffe
28
36
44
52
60
68
76
84
92
00
08
16
24
32
40
48
56

FIG. 11

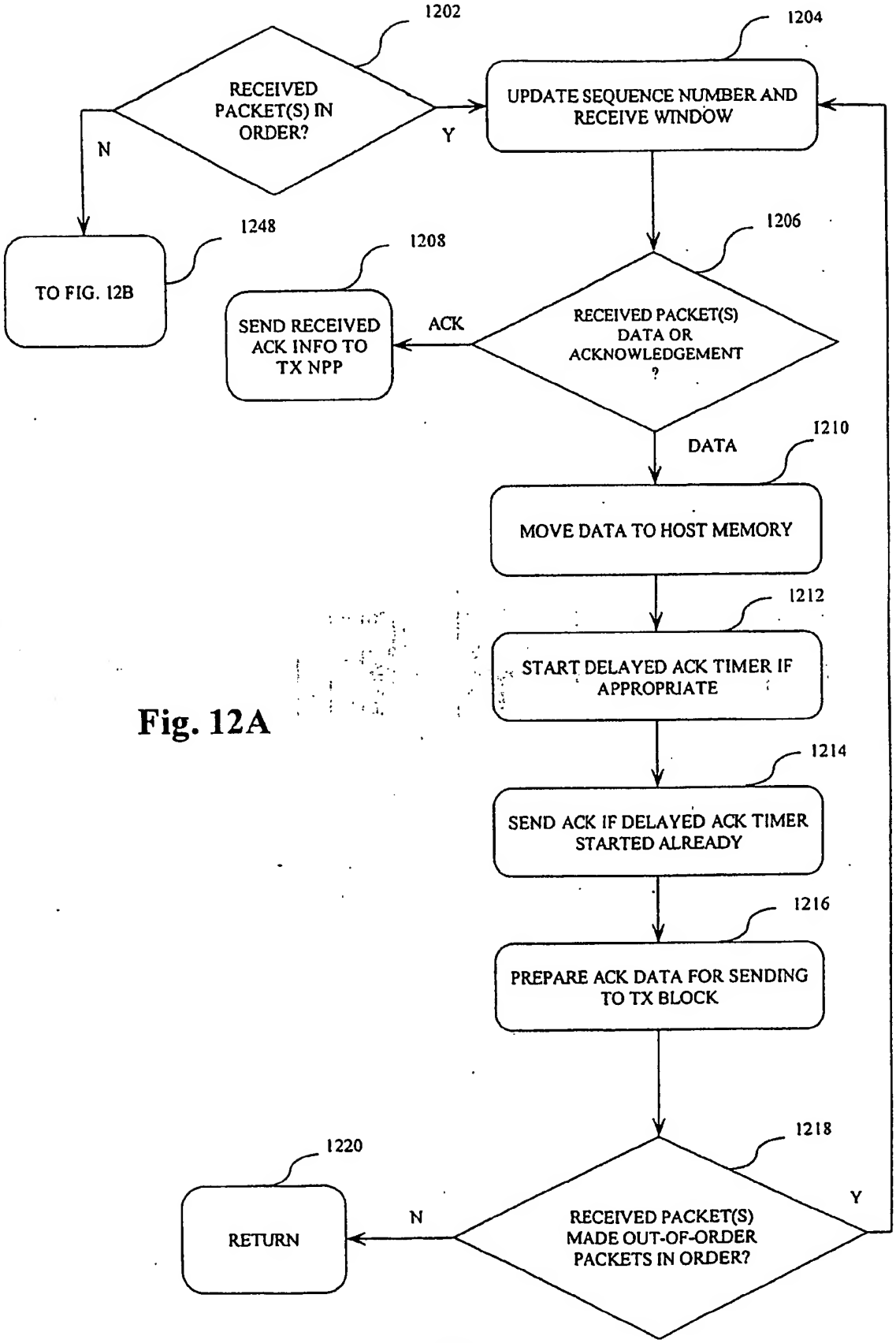

1202
RECEIVED PACKET(S) IN ORDER?
N
Y
1204
UPDATE SEQUENCE NUMBER AND RECEIVE WINDOW
1248
TO FIG. 12B
1208
SEND RECEIVED ACK INFO TO TX NPP
ACK
1206
RECEIVED PACKET(S) DATA OR ACKNOWLEDGEMENT ?
DATA
1210
MOVE DATA TO HOST MEMORY
1212
START DELAYED ACK TIMER IF APPROPRIATE
1214
SEND ACK IF DELAYED ACK TIMER STARTED ALREADY
1216
PREPARE ACK DATA FOR SENDING TO TX BLOCK
1218
RECEIVED PACKET(S) MADE OUT-OF-ORDER PACKETS IN ORDER?
Y
N
1220
RETURN

Fig. 12A

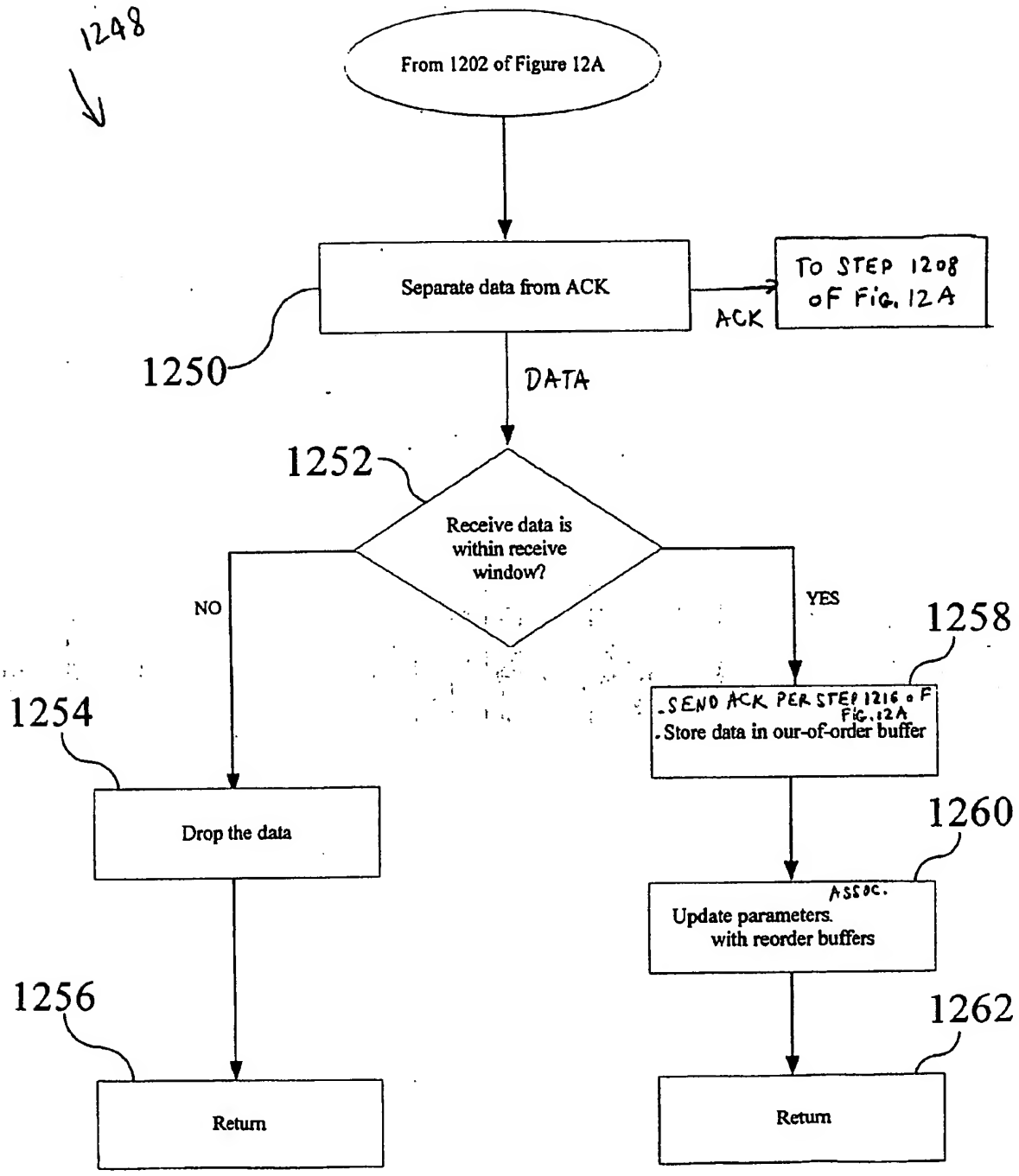

1248
From 1202 of Figure 12A
Separate data from ACK
1250
ACK
TO STEP 1208 OF FIG. 12A
DATA
1252
Receive data is within receive window?
NO
YES
1254
Drop the data
1256
Return
1258
. SEND ACK PER STEP 1216 OF FIG. 12A
. Store data in our-of-order buffer
1260
ASSOC.
Update parameters. with reorder buffers
1262
Return

Fig. 12B

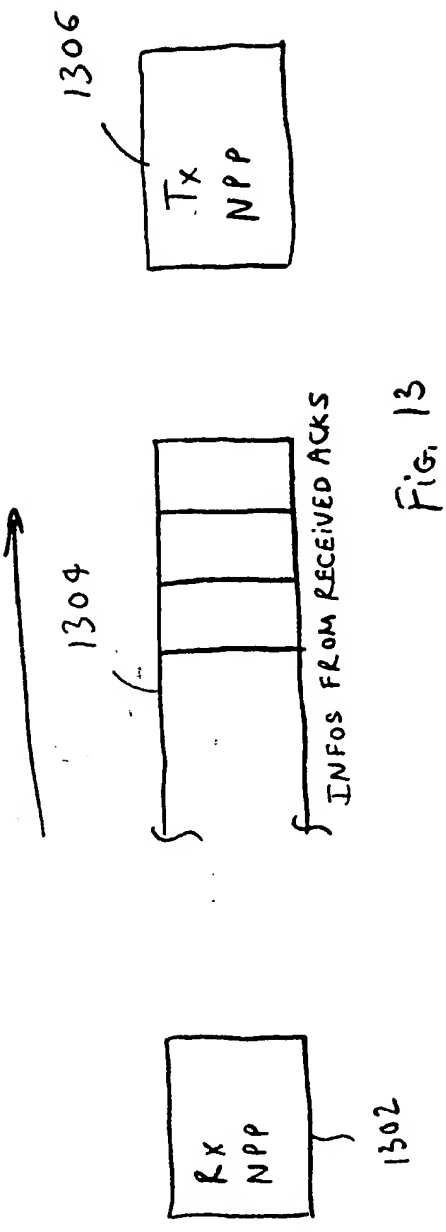

1306
Tx NPP
1304
INFOS FROM RECEIVED ACKS
Rx NPP
1302

Fig. 13

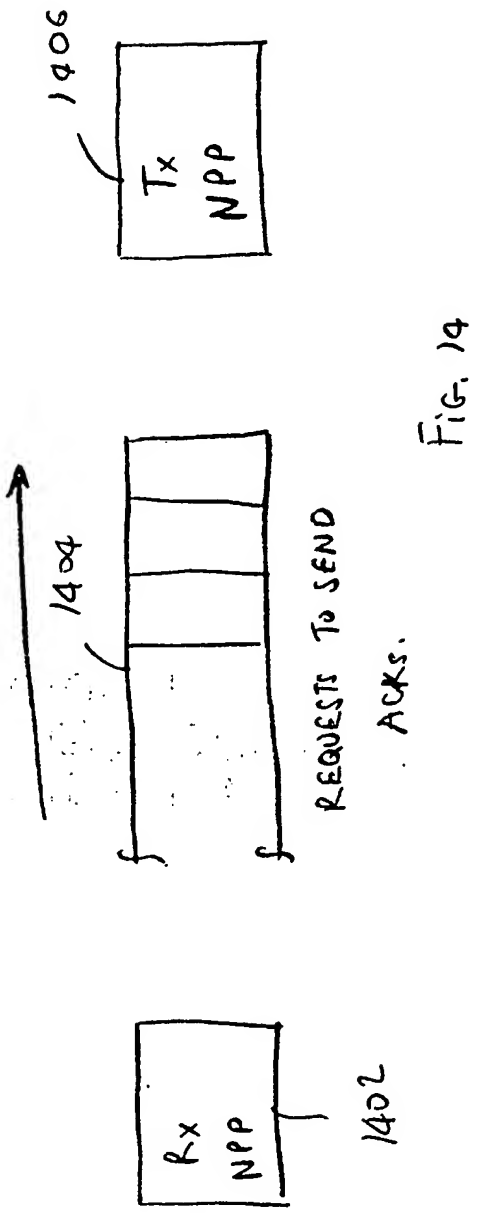

1406
Tx NPP
1404
REQUESTS TO SEND ACKS.
Rx NPP
1402

Fig. 14

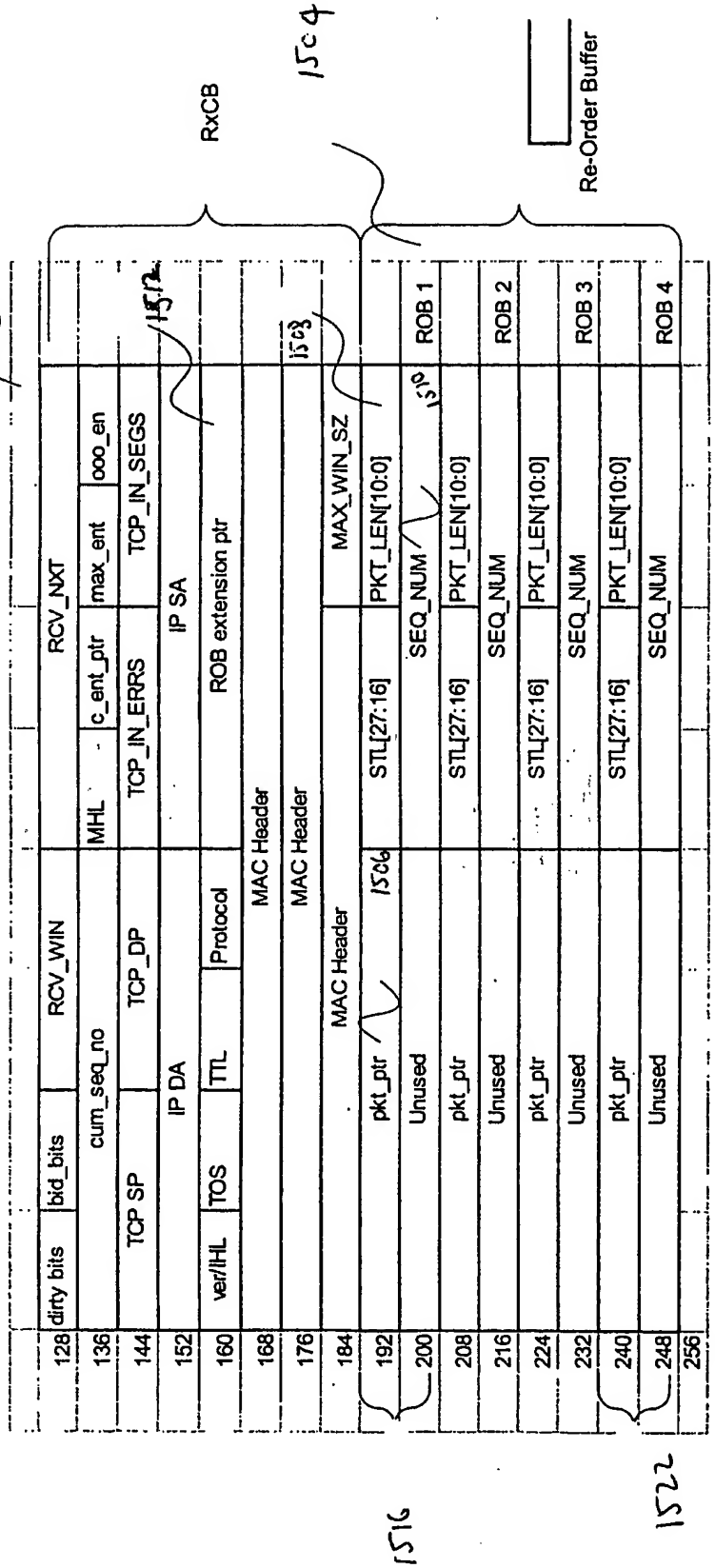

1502
RxCB
1504
Re-Order Buffer
dirty bits
bid_bits
RCV_WIN
RCV_NXT
cum_seq_no
MHL
c_ent_ptr
max_ent
ooo_en
TCP SP
TCP_DP
TCP_IN_ERRS
TCP_IN_SEGS
IP DA
TTL
Protocol
IP SA
ROB extension ptr
ver/IHL
TOS
MAC Header
MAC Header
MAC Header
MAX_WIN_SZ
pkt_ptr
STL[27:16]
PKT_LEN[10:0]
SEQ_NUM
ROB 1
Unused
pkt_ptr
STL[27:16]
PKT_LEN[10:0]
SEQ_NUM
ROB 2
Unused
pkt_ptr
STL[27:16]
PKT_LEN[10:0]
SEQ_NUM
ROB 3
Unused
pkt_ptr
STL[27:16]
PKT_LEN[10:0]
SEQ_NUM
ROB 4
Unused
128
136
144
152
160
168
176
184
192
200
208
216
224
232
240
248
256
1506
1508
1510
1512
1516
1522

FIG. 15

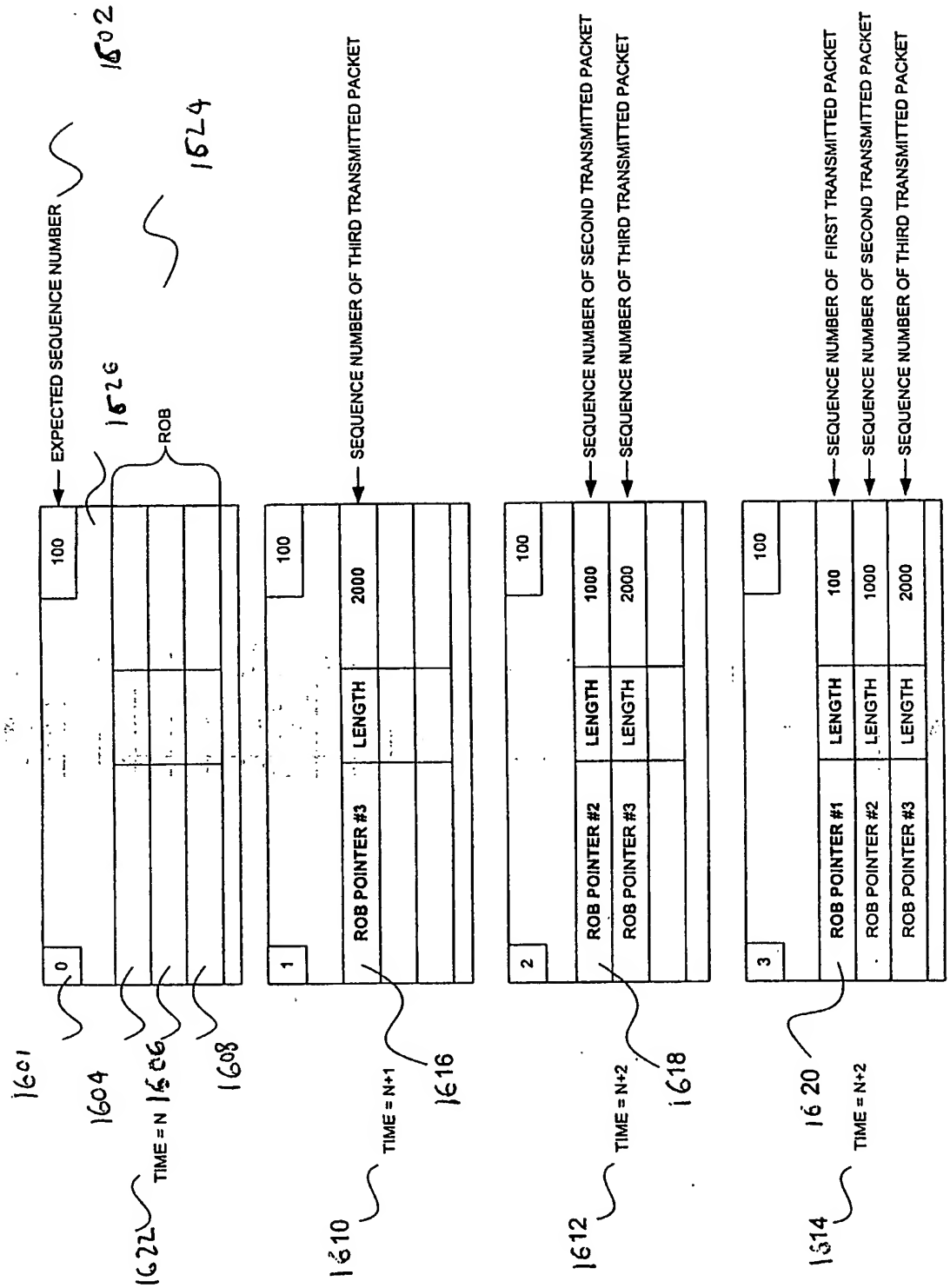

1602
1624
1626
EXPECTED SEQUENCE NUMBER
ROB
SEQUENCE NUMBER OF THIRD TRANSMITTED PACKET
SEQUENCE NUMBER OF SECOND TRANSMITTED PACKET
SEQUENCE NUMBER OF THIRD TRANSMITTED PACKET
SEQUENCE NUMBER OF FIRST TRANSMITTED PACKET
SEQUENCE NUMBER OF SECOND TRANSMITTED PACKET
SEQUENCE NUMBER OF THIRD TRANSMITTED PACKET
0
1
2
3
100
LENGTH
ROB POINTER #1
ROB POINTER #2
ROB POINTER #3
2000
1000
100
1601
1604
1606
1608
1610
1612
1614
1616
1618
1620
1622
TIME = N
TIME = N+1
TIME = N+2
TIME = N+2

Fig. 16

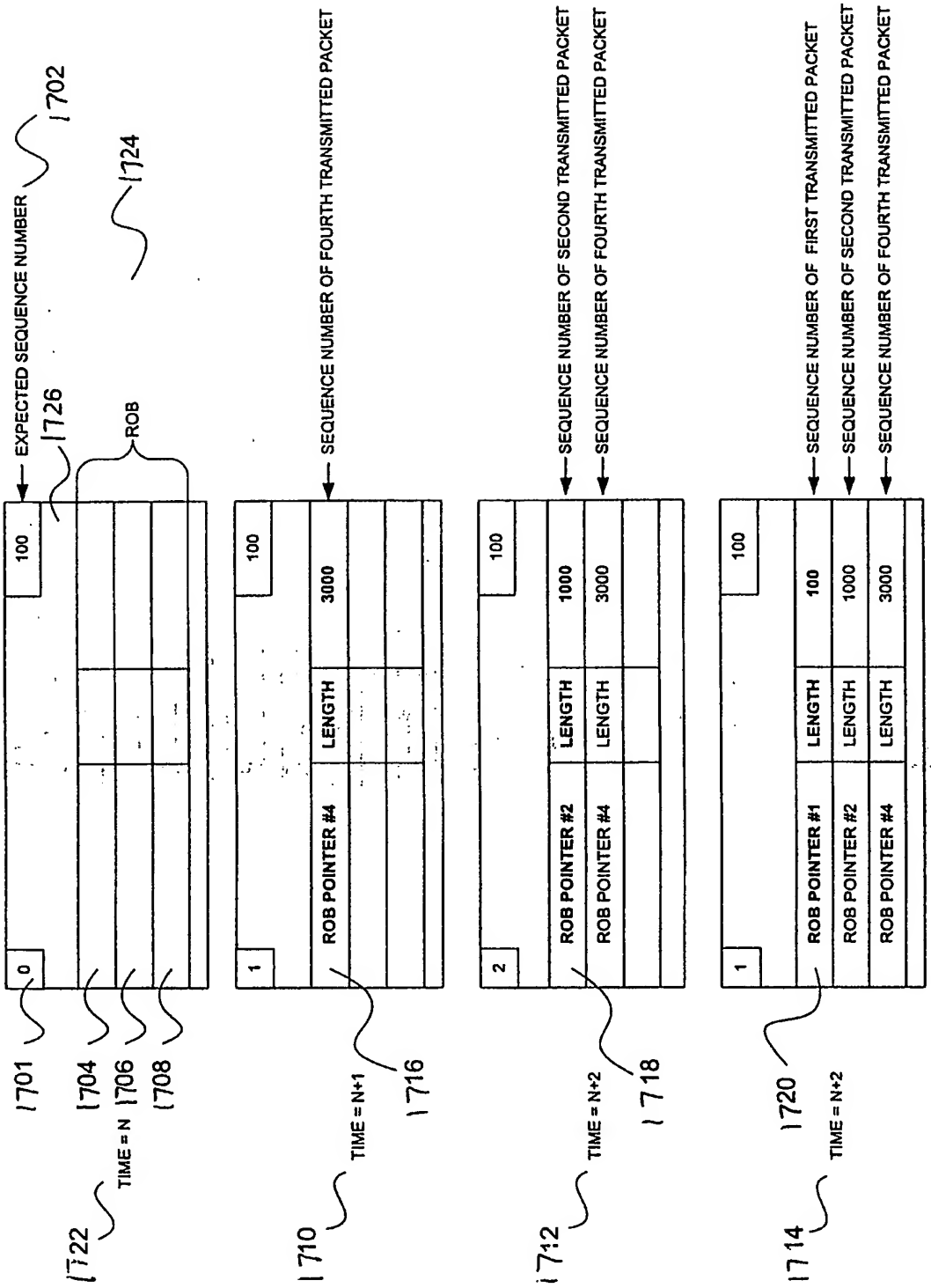

EXPECTED SEQUENCE NUMBER
1702
1724
1726
ROB
100
0
1701
1704
1706
1708
TIME = N
1722
SEQUENCE NUMBER OF FOURTH TRANSMITTED PACKET
100
1
ROB POINTER #4
LENGTH
3000
1716
TIME = N+1
1710
SEQUENCE NUMBER OF SECOND TRANSMITTED PACKET
SEQUENCE NUMBER OF FOURTH TRANSMITTED PACKET
100
2
ROB POINTER #2
ROB POINTER #4
LENGTH
LENGTH
1000
3000
1718
TIME = N+2
1712
SEQUENCE NUMBER OF FIRST TRANSMITTED PACKET
SEQUENCE NUMBER OF SECOND TRANSMITTED PACKET
SEQUENCE NUMBER OF FOURTH TRANSMITTED PACKET
100
1
ROB POINTER #1
ROB POINTER #2
ROB POINTER #4
LENGTH
LENGTH
LENGTH
100
1000
3000
1720
TIME = N+2
1714

Fig. 17

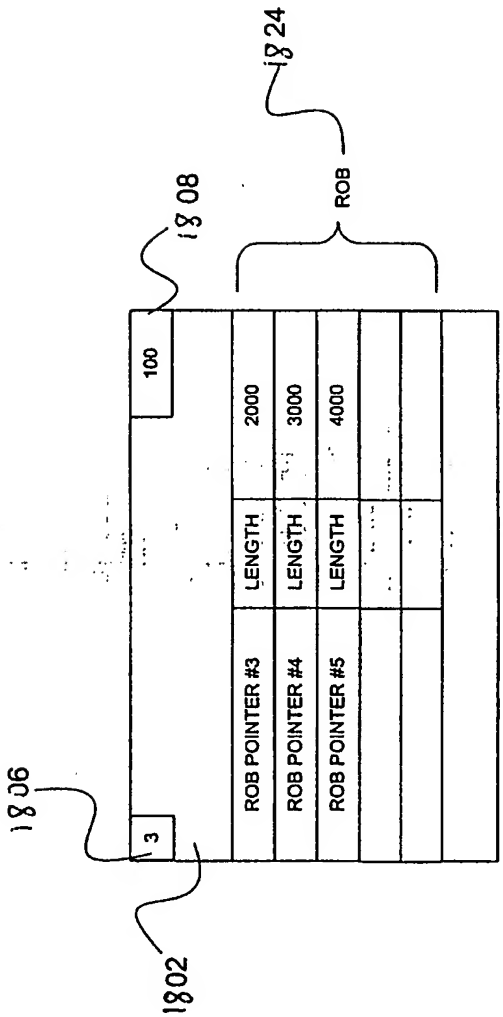

1824
ROB
1808
100
2000
3000
4000
LENGTH
LENGTH
LENGTH
ROB POINTER #3
ROB POINTER #4
ROB POINTER #5
1806
3
1802

Fig. 18

Source Port		Destination Port	
Sequence Number			
Acknowledgement Number			
HLEN	Reserved	Code Bits	Window
Checksum (15-0)		Urgent Pointer	
Options (if any)			Padding
Data			

Fig. A1 Prior Art

TCB LAYOUT IN MEMORY

Gray = TCP Ports, Green = IP Header Related
Blue = Transmit Sequence Variables
Yellow = Receive Sequence Variables
Tan = Re-Order Buffer (ROB)
Red = MAC header, grey = TCP statistics

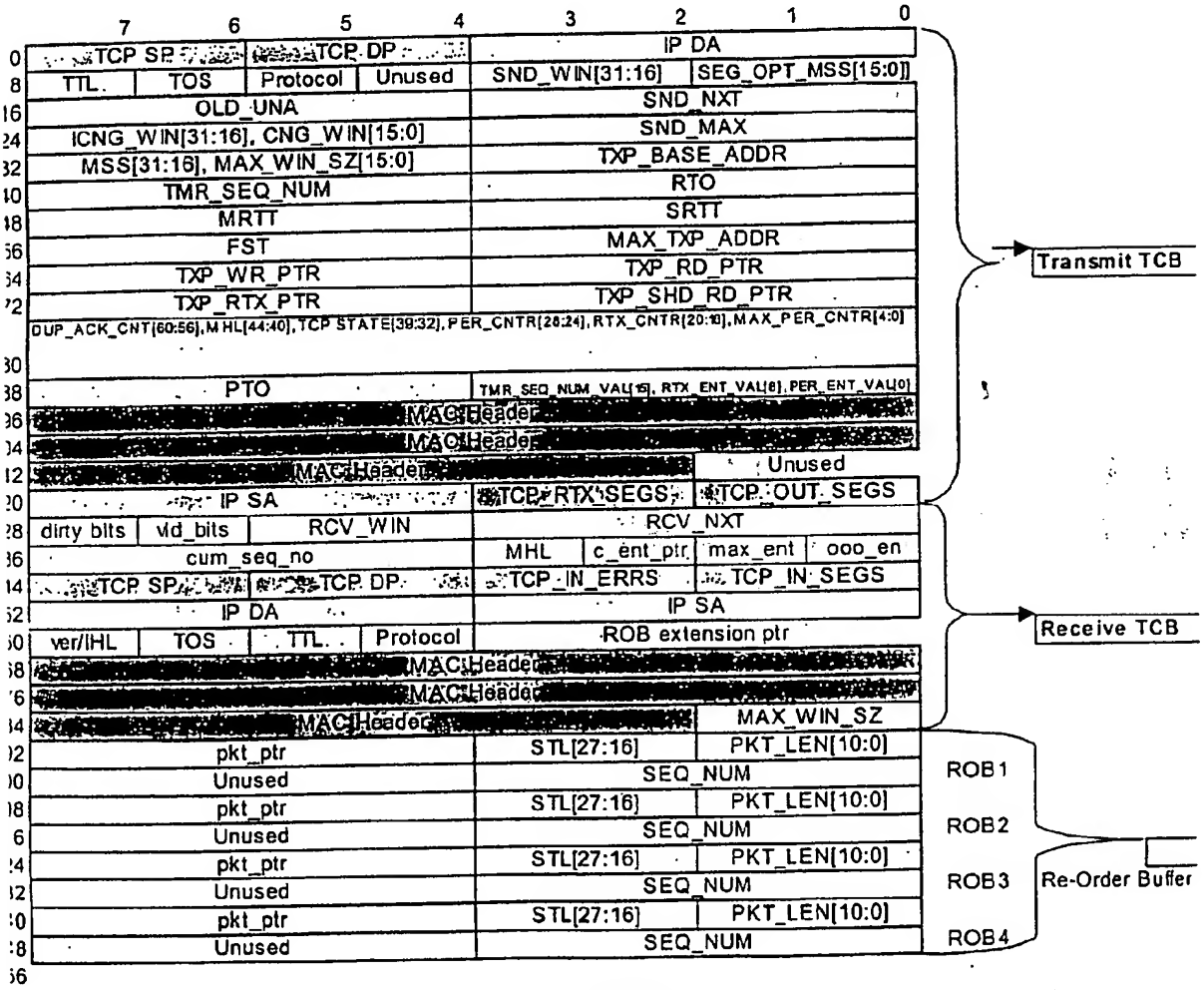


Fig. 2a

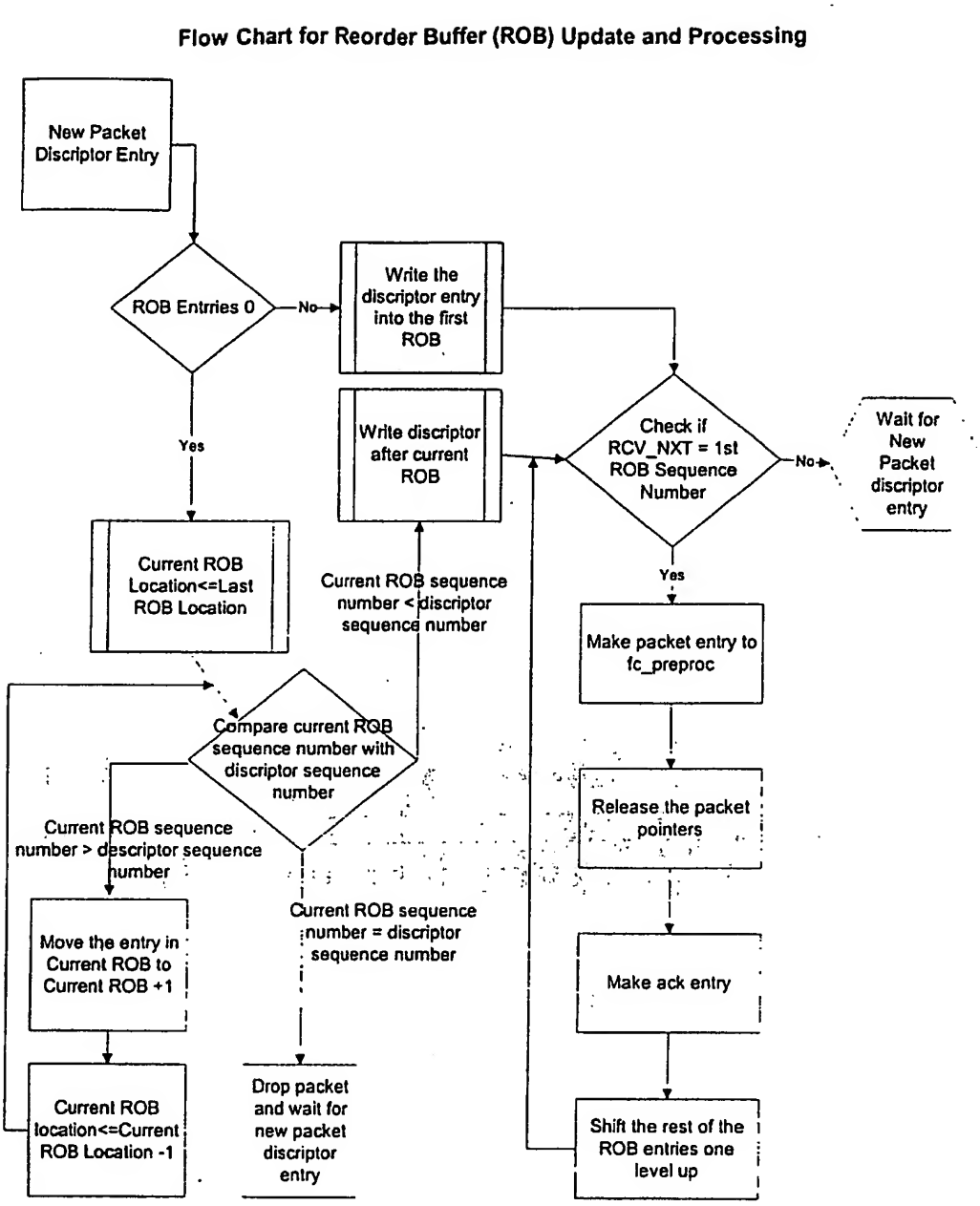

Flow Chart for Reorder Buffer (ROB) Update and Processing
New Packet Discriptor Entry
ROB Entrries 0
No
Yes
Write the discriptor entry into the first ROB
Write discriptor after current ROB
Current ROB Location<=Last ROB Location
Current ROB sequence number < discriptor sequence number
Compare current ROB sequence number with discriptor sequence number
Current ROB sequence number > descriptor sequence number
Move the entry in Current ROB to Current ROB +1
Current ROB location<=Current ROB Location -1
Current ROB sequence number = discriptor sequence number
Drop packet and wait for new packet discriptor entry
Check if RCV_NXT = 1st ROB Sequence Number
No
Wait for New Packet discriptor entry
Yes
Make packet entry to fc_preproc
Release the packet pointers
Make ack entry
Shift the rest of the ROB entries one level up

Fig.A2b

TCB Table

TCP Transmit Variables:

TCB Field	Initial Value	Updated By	Needed by SW TCB	Description
TCP_SP[15:0]	From SW		Yes	TCP Source Port.
TCP_DP[15:0]	From SW		Yes	TCP Destination Port.
IP_DA[31:0]	From SW		Yes	IP Destination Address.
TTL[7:0]	From SW		Yes	Time To Live. Used in IP Header.
TOS[7:0]	From SW		Yes	Type Of Service. Used in IP Header.
PROTOCOL[7:0]	From SW		Yes	Protocol Field. Used in IP Header.
SND_WIN[15:0]	From SW	Tcp_tx_proc, by tcp_rx_ack_proc when an ACK is received.	Yes	This is the window size offered by the sender.
SEG_OPT_MSS	From SW		No	Segment Optimal MSS.
OLD_UNA[31:0]	From SW	Tcp_tx_proc, by tcp_rx_ack_proc when ACK is received	Yes	Oldest Unacknowledged Sequence Number. Same as SND_UNA.
SND_NXT[31:0]	From SW	Tcp_tx_proc, by tcp_norm_cpu_proc, tcp_rx_ack_proc, when entry is put in port_tx_que	Yes	Sequence number to be sent next in the TCP header.
ICNG_WIN[15:0]	From SW		Yes	This is the value used for CNG_WIN when congestion control is used upon retransmission.
CNG_WIN[15:0]	From SW	Tcp_tx_proc, by tcp_rx_ack_proc for slow start on ACK receive, by tcp_rtx_proc when retransmission happens.	Yes	Current congestion window.
SND_MAX[31:0]	From SW	Tcp_tx_proc, by tcp_rx_ack_proc when ACK is received	Yes	Maximum Sequence Number allowed to be sent

FIG 3a

TCB Field	Initial Value	Updated By	Needed by SW TCB	Description
MSS[15:0]	From SW		Yes	Maximum Segment size. This value is aligned to 256-byte count.
MAX_WIN_SZ[15:0]	From SW		Yes	This is the window size which will be advertised as offered window when ACK is sent.
TXP_BASE_ADDR[31:0]	From SW		No	Base address for TX Pending queue. When the pointers roll over, they roll back to this value.
TMR_SEQ_NUM[31:0]	Don't Care	Tcp_tx_proc, when an entry is made in port_tx_que either by tcp_norm_cpu_tx_proc or by tcp_rx_ack_proc or after ACK validation if there are pending entries	No	Sequence number of the segment for which timer measurements are kicked off. The timers are updated when an ACK with ACK_NUM >= TMR_SEQ_NUM is received.
RTO[31:0]	Don't Care	Tcp_tx_proc, by tcp_rx_ack_proc after ACK validation	No	Retransmission Time Out value. It is updated every time an ACK with ACK_NUM >= TMR_SEQ_NUM is received.
MRTT[31:0]	Don't Care	Tcp_tx_proc, by tcp_rx_ack_proc after ACK validation	Yes	Measured Round Trip Time. It is updated every time an ACK with ACK_NUN >= TMR_SEQ_NUM is received.
SRTT[31:0]	From SW (In terms of number of 125 Mhz cycles.)	Tcp_tx_proc, by tcp_rx_ack_proc after ACK validation	Yes	Smoothened Round Trip Time. It is updated every time an ACK with ACK_NUN >= TMR_SEQ_NUM is received.

FIG.3b

TCB Field	Initial Value	Updated By PCT Need	d by SW TCB	Description 09
FST[31:0]	Don't Care	Tcp_tx_proc, when an entry is made in port_tx_que either by tcp_norm_cpu_tx_proc or by tcp_rx_ack_proc	No	Frame Transmit Time. The current time at which the frame entry was made in the port_tx_que. Obtained from a 32-bit counter.
MAX_TXP_ADDR[31:0]	From SW		No	Maximum address for TX Pending queue. When TXP pointers are greater than this address, they roll back to the starting address.
TXP_WR_PTR[31:0]	From SW (beginning of TXP queue)	Tcp_tx_proc, by tcp_norm_cpu_proc after an entry is serviced from pre_tx_que.	No	Current Value of TX Pending queue write pointer.
TXP_RD_PTR[31:0]	From SW (beginning of TXP queue)	Tcp_tx_proc, by tcp_norm_cpu_proc or tcp_rx_ack_proc	No	Current value of TX Pending queue read pointer.
TXP_RTX_PTR[31:0]	From SW (beginning of TXP queue)	Tcp_tx_proc, by tcp_rx_ack_proc	No	Current value of the TX Pending retransmit pointer. On a time out, entry pointed by RTX pointer is processed.
TXP_SHD_RD_PTR[31:0]	From SW (beginning of TXP queue)	Tcp_tx_proc, by tcp_norm_cpu_proc, tcp_rtx_proc	No	Current value of the TX Pending queue shadow read pointer. Used during retransmission of frames.
DUP_ACK_CNT[4:0]	From SW (zero)	Tcp_tx_proc, tcp_rx_ack_proc	Yes	Current value of duplicate ACK count.
MHL[4:0]	From SW			MAC Header Length.
TCP_TX_ST[3:0]	From SW	TBD	Yes	Current TCP transmit state.
TCP_RX_ST[3:0]	From SW	TBD	Yes	Current TCP receive state.
RTX_CNT[4:0]	From SW (zero)	Tcp_tx_proc, tcp_rtx_proc, tcp_rx_ack_proc	Yes	Current value of Retransmit counter.

FIG.3c

TCB Field	Initial Value	Updated By	Needed by SW TCB	Description
PTO[31:0]	From SW (In terms of number of 125 Mhz cycles.)	Tcp_tx_proc, tcp_tx_pst_proc after sending a persist probe. If an ACK for the probe is not received and timeout occurs OR if an ACK with zero window size is received, then the persist probe is sent again and this time is doubled until MAX_PTO is reached. Then it maintains MAX_PTO value. It is reinitialized to INIT_PTO when an ACK with non-zero offered window size is received.	Yes	Persist Time Out value.
TMR_SEQ_NUM_VAL	From SW (zero)	Tcp_tx_proc. If reset, this bit is set by tcp_tx_norm_cpu_proc. Reset by tcp_rx_ack_proc when TX Pending queue is empty.	No	Timer Sequence Number Valid. When an ACK with ACK_NUM >= TME_SEQ_NUM is received, timer calculations are done. Also, if there are pending entries in the TXP, a new timeout value in written into RTX SRAM. When an ACK is received and there is nothing in the TX Pending queue, then this bit is reset.
RTX_ENT_VAL	From SW (zero)	Tcp_tx_proc, If this bit is reset, then it is set by the process making an entry in the port_tx_que.	No	Retransmit entry valid. When set, it means that the RTX timer is running.
PST_ENT_VAL	From SW (zero)	Tcp_tx_proc, by tcp_rx_ack_proc. Set when an ACK with zero offered window is received. Reset when an ACK with non-zero offered window is received.	No	Persist Entry Made.

FIG.3d

TCB Field	Initial Value	Updated By	Needed by SW TCB	Description
MAC[191:0]	From SW		No	MAC Header. Maximum possible header length is 22-bytes. The actual length is given by MHL. This field appears twice so it can be accessed by the RX engine for ougoing ack packets.
TCP_RTX_SEGS[15:0]	From SW	Tcp_tx_proc, tcp_rtx_proc when a segment is retransmitted. Need to calculate the segmented packets.	Yes	Total number of segments retransmitted. (takes into account segmentation).
TCP_OUT_SEGS[15:0]	From SW	Tcp_tx_proc, by tcp_nom_cpu_tx_proc or tcp_rx_ack_proc or by tcp_rtx_proc or by tcp_tx_pst_proc, any time an entry is made in the port_tx_queue. Need to calculate the total segmented packets.	Yes	Total number of TCP segments sent out on this session including the segmented packets.

Receive Variables:

TCB Field	Initial Value	Updated By	Needed By SW TCB	Description
RCV_WIN[15:0]	From SW	Tcp_rx_proc	Yes	Current Receive window. When tcp_rx_proc sends an ACK, this value is used in the offered window field in the TCP Header.
RCV_NXT[31:0]	From SW	Tcp_rx_proc	Yes	The next expected sequence number by tcp_rx_proc.
cum_seq_no[31:0]	From SW (= RCV_NXT)	Tcp_rx_proc	No	Accumulated Segment Length. Used in frame reassembly.
vld_bits[7:0]]	From SW	Tcp_rx_proc	No	Valid 64-byte segments in the TCB, each of the 8

FIG 3e

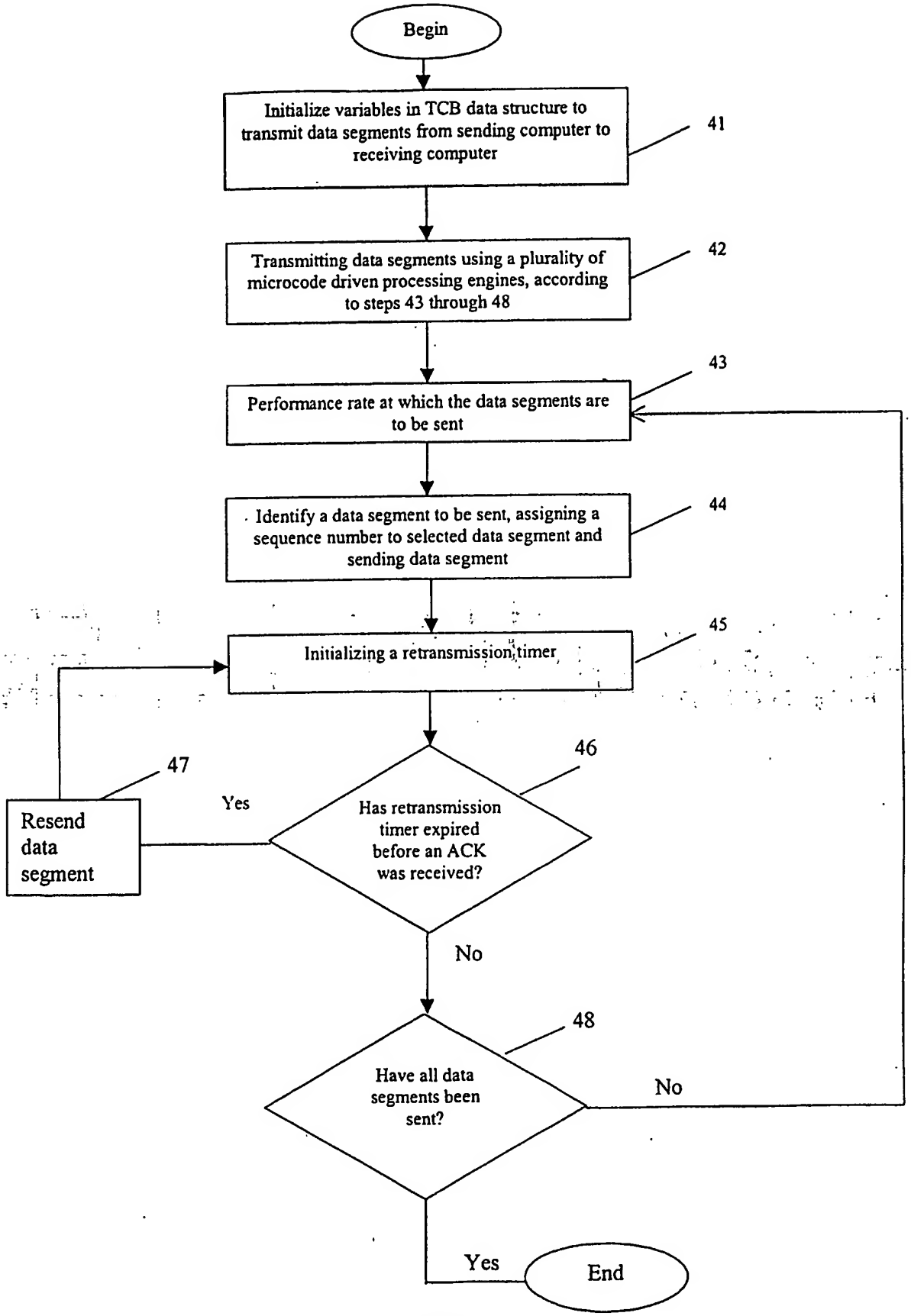
TCB Field	Initial Value	Updated By	Needed By SW TCB	Description
	(0x04)			64-byte segments has a corresponding bit.
dirty_bits[7:0]	From SW (0x00)	Tcp_rx_proc	No	For each of the 8 64-byte TCB/ROB segments a corresponding dirty bit indicates it has been updated and needs to be flushed back to memory..
OOO_EN	From SW	SW	No	When enabled, out of order packets are processed, other wise only sequential packets are processed.
max_ent[3:0]]	From SW	Tcp_rx_proc	No	maximum no. of entries in the ROB, can be upto 20
c_ent_ptr[3:0]	From SW (0x0)	Tcp_rx_proc	No	current entry pointer. Points to next entry to be filled.
TCP_IN_SEGS[15:0]	From SW (0x0)	Tcp_rx_proc	No	current no. of TCP segments received so far.
TCP_IN_ERRS[15:0]	From SW (0x0)	Tcp_rx_proc	No	current no. of TCP segments with errors received so far.
MAC Header	From SW	SW	No	MAC header for use by outgoing ack packet
MHL	From SW	SW	No	MAC Header Length for use by the outgoing ack packet
TCP_SP[15:0]	From SW		Yes	TCP Source Port.
TCP_DP[15:0]	From SW		Yes	TCP Destination Port.
MAX_WIN_SZ[15:0]	From SW		Yes	This is the window size which will be advertised as offered window when ACK is sent.
TTL[7:0]	From SW		Yes	Time To Live. Used in IP Header.
TOS[7:0]	From SW		Yes	Type Of Service. Used in IP Header.
PROTOCOL[7:0]	From SW		Yes	Protocol Field. Used in IP Header.

FIG.13f

TCB Field	Initial Value	Updated By	Needed By SW TCB	Description
IP_SA[31:0]	From SW		Yes	IP Source Address.
IP_DA[31:0]	From SW		Yes	IP Destination Address.

Re-Order Buffer:

TCB Field	Initial Value	Updated By	Needed By SW TCB	Description
FRM_LEN[15:0]	Don't Care	Tcp_rx_proc when out of order packets are received.	No	Frame length for the out of order segment.
PKT_LEN[15:0]	Don't Care	Tcp_rx_proc when out of order packets are received.	No	Length of the current segment.
SEQ_NUM[31:0]	Don't Care	Tcp_rx_proc when out of order packets are received.	No	Sequence Number of out of order segment.
SAF[31:0]	Don't Care	Tcp_rx_proc when out of order packets are received	No	Start Address where the out of order segment is stored in SDRAM1.


Begin
Initialize variables in TCB data structure to transmit data segments from sending computer to receiving computer
41
Transmitting data segments using a plurality of microcode driven processing engines, according to steps 43 through 48
42
Performance rate at which the data segments are to be sent
43
Identify a data segment to be sent, assigning a sequence number to selected data segment and sending data segment
44
Initializing a retransmission timer
45
Has retransmission timer expired before an ACK was received?
46
Yes
Resend data segment
47
No
Have all data segments been sent?
48
No
Yes
End

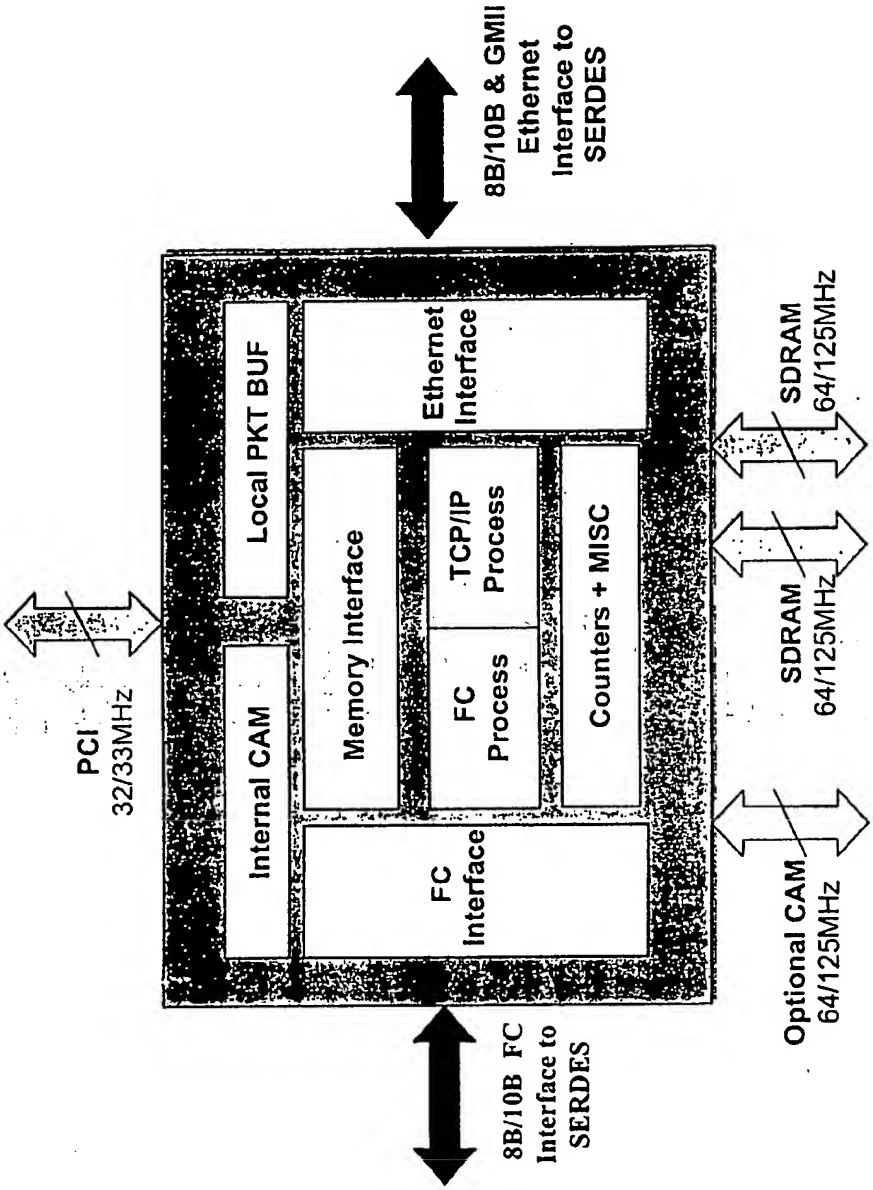
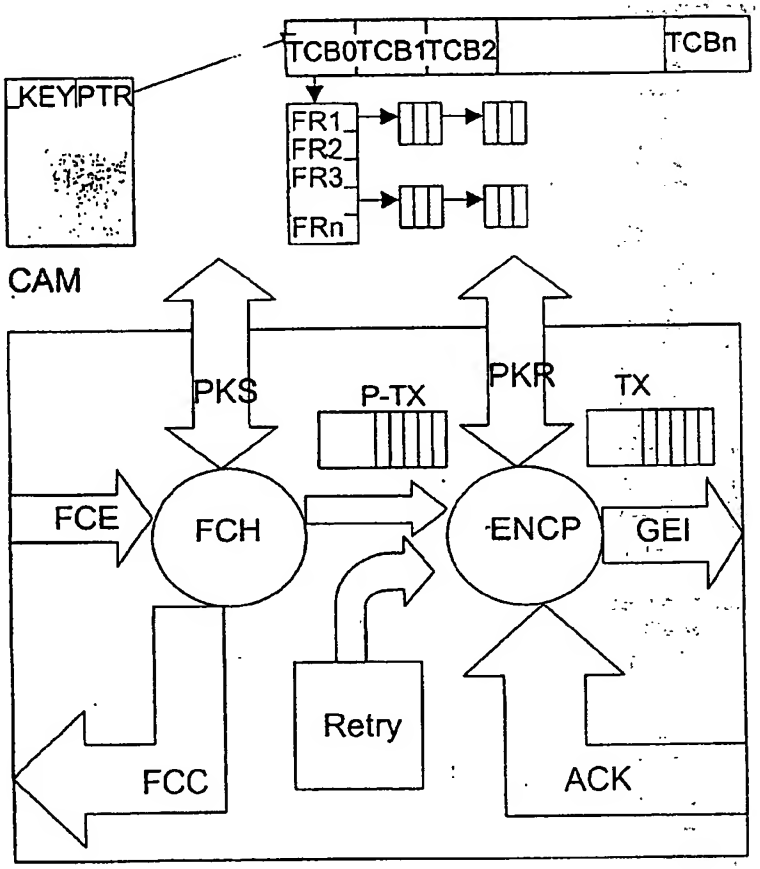

8B/10B & GMII Ethernet Interface to SERDES
Local PKT BUF
Ethernet Interface
Memory Interface
TCP/IP Process
FC Process
Counters + MISC
Internal CAM
FC Interface
PCI 32/33MHz
SDRAM 64/125MHz
SDRAM 64/125MHz
Optional CAM 64/125MHz
8B/10B FC Interface to SERDES

Fig. A5

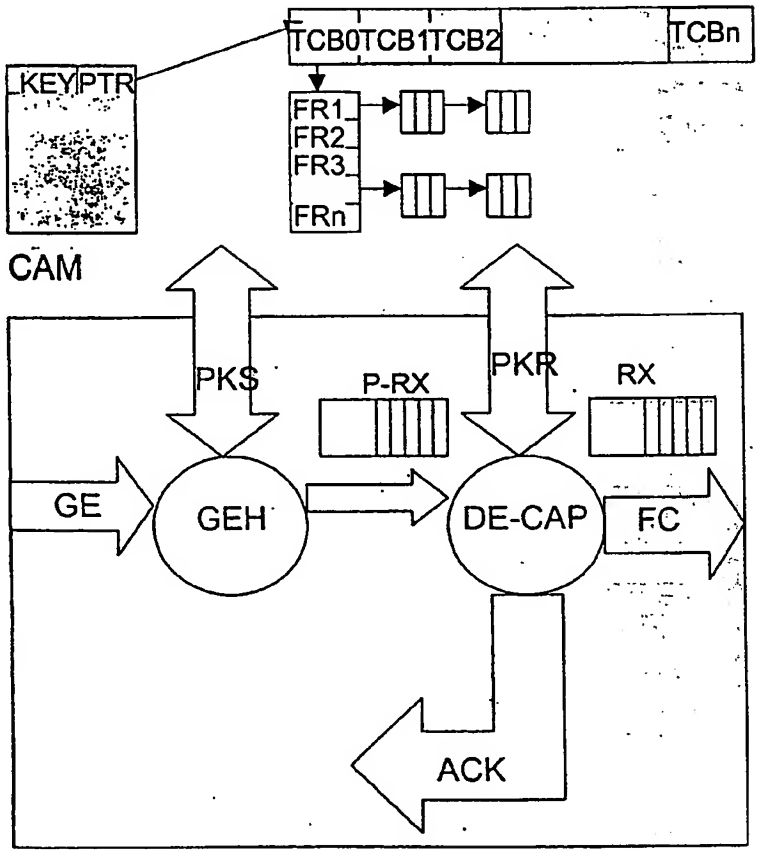
FC to GE Processing



- FCE - FCH - PKS
  - Fibre channel engine, forwards the frames to FC header processing engine (NPP1), and stores the frame in memory (TX-Pending per TCB)
- CNTL-ENCP-GEI
  - Encapsulation engine (NPP2), prepares the STP/TCP/IP header for every frame. Ethernet framing and segmentation are also done in this block
- Retry
  - Identifies and queues TX frames for retransmit. TCP Timers on chip
- ACK-FCC
  - Sliding window process and FC-credit release on acknowledge frames

Fig. A6

GE to FC Processing



- GE-GEH
  - Gigabit MAC, forwards the frames to Gigabit header processing engine (NPP3), and stores the frame in memory (RX-Pending per TCB). Fetches the TCB pointer using CAM
- DE-CAP - FC
  - De-capsulation and re-assembly engine (NPP4). Removes STP/TCP/IP header for every frame and re-assembles. Window is acknowledged for good TCP packets, thereby advancing TX window.
- FC
  - FC engine transmits the frames to FC line

Fig. A7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/27709

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 12/00; H04L 12/56
US CL : 712/300; 370/389, 394
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 712/300; 370/389, 394

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 6,393,023 B1 (SHIMIZU et al) 21 May 2002, see whole document.	1-12
Y, E	US 6,457,121 B1 (KOKER et al) 24 September 2002, see whole document.	1-12
Y	US 6,246,684 B1 (CHAPMAN et al) 12 June 2001, see whole document.	1-12

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 September 2002 (30.09.2002)	05 NOV 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer for James R. Matthews Ayaz R. Sheikh Telephone No. 703-305-9648

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US02/27709

Continuation of B. FIELDS SEARCHED Item 3:

East search key terms:

transmitted device, source device, receiving device, target device, reordering data packets, transmit data packet, memory buffer, storing data packet